Marie-Aude Aufaure
Esteban Zimányi (Eds.)

Tutorial
LNBIP 96

Business Intelligence

First European Summer School, eBISS 2011
Paris, France, July 2011
Tutorial Lectures

Springer

Lecture Notes
in Business Information Processing 96

Marie-Aude Aufaure
Esteban Zimányi (Eds.)

Business Intelligence

First European Summer School, eBISS 2011
Paris, France, July 3-8, 2011
Tutorial Lectures

Springer

Volume Editors

Marie-Aude Aufaure
Ecole Centrale Paris
MAS Laboratory
Châtenay-Malabry Cedex, France
E-mail: marie-aude.aufaure@ecp.fr

Esteban Zimányi
Universite Libre de Bruxelles
Department of Computer & Decision Engineering (CoDE)
Brussels, Belgium
E-mail: ezimanyi@ulb.ac.be

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-642-27357-5 e-ISBN 978-3-642-27358-2
DOI 10.1007/978-3-642-27358-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011944231

ACM Computing Classification (1998): J.1, H.2, H.3, D.1

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The First European Business Intelligence Summer School (eBISS 2011) attracted experts from academia and industry as well as PhD students interested in foundational and applicational aspects of business intelligence (BI). This volume contains the lecture notes of the summer school, which took place in Paris, France, in July 2011.

The first chapter reviews consolidated results in data warehouses and describes open research fields. The latter include the need to cope with more complex data, both in structure and semantics, and keeping up with the demands of new application domains such as Web, financial, manufacturing, life sciences, multimedia, and spatiotemporal applications.

The second chapter delves into the issue of data warehouse performance. It reviews three types of data structures, namely, indexes, materialized views, and partitioned tables, which are essential for efficiently answering analytical queries. The chapter also shows how these techniques are applied when executing star queries in three commercial data warehouse systems.

The third chapter shows how popular user-centric techniques, namely, personalization and recommendation, can be applied to OLAP queries. The chapter characterizes the approaches proposed in this respect in terms of formulation effort, prescriptiveness, proactiveness, expressiveness, and in terms of the data leveraged.

The fourth chapter discusses the issue of supporting analytical queries over Web-based textual content. This problem lies at the intersection of the domains of search engines and BI. The chapter shows three recent enabling technologies in this respect, namely, cloud computing, self-supervised keyword generation, and fact extraction. The chapter ends by describing the GoOLAP system, a platform supporting Web-scale business analytics.

The fifth chapter provides an overview of Business Intelligence 2.0, an extension of traditional BI based on the evolution of the Web and emerging technologies such as cloud computing. BI 2.0 promises to enable better decision making by complementing traditional organizational data with information present in the Web, like opinions or information about competitors,

while using collective intelligence, collaborative work through social networks, and supporting BI systems with cloud computing.

The sixth chapter surveys the issue of graph mining over social networks. In this respect, community detection may bring very valuable information about the structure of an existing social network. The chapter defines what is a community in a social network context, and then surveys the most popular techniques to detect such communities.

The seventh chapter presents an overview of the Semantic Web, with a special focus on semantic databases, or triplestores, which are specific databases aimed at integrating, storing, and querying the vast amounts of data generated on the Semantic Web every day. The chapter also provides an overview of BI-related scenarios where semantic technologies and triplestores provide valuable advantage and differentiation.

The eight chapter introduces the service paradigm and analyzes its impacts on BI. Specific techniques to engineering service systems are presented, including cloud computing, service-oriented architectures (SOA), and business process modeling (BPM). The chapter also analyzes whether it is possible to consider BI as a service, and analyzes how to use BI techniques to enhance services.

Finally, the ninth chapter explores collaborative BI, which aims at extending the decision-making process beyond the company boundaries. Due to its inherent distribute nature, collaborative BI requires innovative approaches and architectures. The chapter surveys data warehouse integration as an enabling technique for collaborative BI and outlines a new peer-to-peer framework, called Business Intelligence Network, aiming at sharing business information for the decision-making process.

The lectures of the summer school surveyed established areas in the BI domain. They also also pointed out aspects that are beginning to be explored or are still waiting for a solution. We hope that the school's material will inspire further exciting research in these areas. We are grateful to all the lecturers and their co-authors for their excellent contributions, the participants of the summer school for their enthusiasm, and the external referees for their careful work that helped us to improve the lectures in this volume.

October 2011

Marie-Aude Aufaure
Esteban Zimányi

Organization

The First European Business Intelligence Summer School (eBISS 2011) was organized by the MAS Laboratory of the Ecole Centrale de Paris and the Department of Computer and Decision Engineering (CoDE) of the Université Libre de Bruxelles.

Program Committee

Alberto Abelló	Universitat Politècnica de Catalunyia, Spain
Marie-Aude Aufaure	Ecole Centrale de Paris, France
Patrick Marcel	Université François Rabelais de Tours, France
Alexander Löser	Technische Universität Berlin, Germany
Esteban Zimányi	Université Libre de Bruxelles, Belgique

Local Organizers

Marie-Aude Aufaure	Ecole Centrale de Paris, France
Nesrine Ben Mustapha	Ecole Centrale de Paris, France
Etienne Cuvelier (Organization Chair)	Ecole Centrale de Paris, France
Micheline Elias	Ecole Centrale de Paris, France
Nicolas Kuchmann-Beauger	Ecole Centrale de Paris, France
Cassio Melo	Ecole Centrale de Paris, France
Nizar Messai	Ecole Centrale de Paris, France
Vincent Mousseau	Ecole Centrale de Paris, France
Rania Soussi	Ecole Centrale de Paris, France
Esteban Zimányi	Université Libre de Bruxelles, Belgium

External Referees

Alfredo Cuzzocrea	ICAR-CNR and University of Calabria, Italy
Cécile Favre	Université Lyon 2, France
Matteo Golfarelli	University of Bologna, Italy
Adriana Marotta	Universidad de la República, Uruguay
Anisoara Nica	SQL Anywhere, Canada
Carlos Ordonez	University of Houston, USA
Raul Ruggia	Universidad de la Republica, Uruguay
Jens Otto Sørensen	Energinet.dk, Denmark
Christian Thomsen	Aalborg University, Denmark
Stijn Vansummeren	Université Libre de Bruxelles, Belgium

Contents

1

Data Warehouses: Next Challenges

Alejandro Vaisman and Esteban Zimányi

Department of Computer and Decision Engineering (CoDE)
Université Libre de Bruxelles
{avaisman,ezimanyi}@ulb.ac.be

Summary. Data Warehouses are a fundamental component of today's Business Intelligence infrastructure. They allow to consolidate heterogeneous data from distributed data stores and transform it into strategic indicators for decision making. In this tutorial we give an overview of current state of the art and point out to next challenges in the area. In particular, this includes to cope with more complex data, both in structure and semantics, and keeping up with the demands of new application domains such as Web, financial, manufacturing, genomic, biological, life science, multimedia, spatial, and spatiotemporal applications. We review consolidated resaerch in spatio-temporal databases, and open research fields, like real-time Business Intelligence and Semantic Web Data Warehousing and OLAP.

Keywords: data warehouses, OLAP, spatiotemporal data warehouses, real-time data warehouses, semantic data warehouses.

1.1 Introduction: Data Warehousing Past, Present, and New Applications

OLAP (On-Line Analytical Processing) [1] comprises a set of tools and algorithms that allow efficiently querying multidimensional databases containing large amounts of data, usually called Data Warehouses. In OLAP, data are organized as a set of *dimensions* and *fact tables*. In this multidimensional model, data can be perceived as a *data cube*, where each cell contains measures of interest. OLAP dimensions are further organized in hierarchies that favor the data aggregation process [2]. Several techniques have been developed for query processing, most of them involving some kind of aggregate precomputation [3].

OLAP (or, more generally, Business Intelligence) (BI) software, produces reports and interactive interfaces that summarize data via basic aggregation functions (e.g., counts and averages) over various hierarchical breakdowns of the data into groups, defined in the dimension hierarchies. A lot of academic research and industrial development was carried out throughout the 1990's

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 1–26, 2012.

related to conceptual modeling, query processing and optimization, aggregate precomputation, etc.

1.1.1 New BI Domains and Challenges

Since the mid 90's, DW and BI applications have been built to consolidate enterprise business data, allowing taking timely and informed decisions based on up-to-date consolidated data. However, the availability of enormous amounts of data from different domains is calling for a shift in the way DW and BI practices are being carried out. It is becoming clear that, for certain kinds of BI applications, the traditional approach, where day-to-day business data produced in an organization is collected in a huge common repository for data analysis, needs to be revised, to account for efficiently handling large-scale data. Moreover, in the emerging domains where BI practices are gaining acceptance, massive-scale data sources are becoming common, posing new challenges to the DW research community.

New database architectures are gaining momentum. Parallelism is becoming a must for large DW processing, as Stonebraker states [4]. Moreover, column stores databases are strong candidates for DW architectures, since they deliver much better performance than classic row databases, for fact tables with a large number of attributes. The MapReduce model [5] is becoming increasingly popular, challenging traditional parallel DBMS architectures [6]. Even if it is not clear if this approach can be applied to all kinds of DWs and BI applications (since it has been argued that MapReduce is not appropriate for analyzing structured data, given the typically high number of joins required), many large DW have been built based on this model. Moreover, there is a wave of opinion arguing that data warehousing can take advantage of the scalability of MapReduce (at the low price of adding commodity computer nodes) [7]. As an example, we can mention the (now famous) Facebook data warehouse built using Hadoop (an open source implementation of MapReduce) [8, 9]. Along these same lines, Cohen et al. [10] propose a paradigm shift in DW design, introducing the MAD Skills approach, aimed at allowing more flexibility when selecting the data sources, and more complex analysis functions than the typical OLAP operations.

The above is not the only challenge for OLAP in the years to come. There is also a need to cope with complex data, both in structure and semantics, and keeping up with the demands of new application domains such as Web, financial, manufacturing, genomic, biological, life science, multimedia, spatial, and spatiotemporal applications. Most of these new kinds of data also require the vision shift expressed above, since large amounts of data populate the databases of applications in these domains.

The intention of this document is two-fold: on the one hand, to review the state-of-the-art in the above mentioned domains. On the other hand, to discuss what is needed to be done to address new challenges in the BI world. For space limitations, and to make this discussion more concrete, in the

remainder we focus in three problems: spatio-temporal DW and BI (Section 1.2), Real-Time DW (Section 1.3), and DW on the Semantic Web (Section 1.4). The rationale for this decision is that the first topic has been attracting the attention from research and industry since long ago, and is now quite mature and steadily producing relevant results, some of which we discuss in this work. Real-time DW is a clear challenge, in light of the large volumes of data handled by the organizations, although in our opinion, the field is open for research. Finally, the so-called Semantic Web, where billions of triples are stored daily, will be most likely the place where data and OLAP-style analysis will occur in the future. The analysis of web data (e.g., in RDF [11] format) embeds the problems described above: large amounts of data, volatile sources, and real-time analysis needs.

1.2 Spatio-Temporal Data Warehousing and OLAP

Geographic Information Systems (GIS) have been extensively used in various application domains, ranging from economical, ecological, and demographic analysis, to city and route planning [12]. Spatial information in a GIS is typically stored in different so-called *thematic layers* (or *themes*). Information in themes consists of spatial data (i.e., geometric objects) associated to thematic (alphanumeric) information.

Rivest *et al.* [13] introduced the notion of SOLAP (standing for Spatial OLAP), a paradigm aimed at exploring spatial data by drilling on maps, as it is performed in OLAP with tables and charts. They describe, in an informal way, the desirable features and operators a SOLAP system should have. In one of the first implementation efforts, Shekhar *et al.* [14] introduced MapCube, a visualization tool for spatial data cubes. Given a so-called base map, cartographic preferences, and an aggregation hierarchy, the MapCube operator produces an album of maps that can be navigated via roll-up and drill-down operations. Since then, SOLAP concepts and operators have been included in many commercial tools. Gómez *et al.* [15] introduces a SOLAP system denoted Piet. This proposal in based on a data model where GIS and warehouse data are maintained separately, and a matching function binds the two components. Piet comes equipped with an SQL-like query language, called Piet-QL. This language allows expressing GIS queries filtered by the values in the cells of a data cube (i.e., filtered by aggregated data)[1], and OLAP queries filtered by conditions over spatial data

1.2.1 A Taxonomy for Spatio-Temporal OLAP

We will see in the remainder of this paper, that existing proposals for spatial data warehousing cover different functional requirements, but, with limited

[1] A Piet-QL demo can be found at `http://piet.exp.dc.uba.ar/pietql`

exceptions, there is no clear specification of the kinds of queries these proposals address. The first attempt to provide a formal characterization of spatio-temporal data warehousing is the work by Vaisman and Zimányi [16], where a taxonomy of spatio-temporal OLAP queries is proposed (see Figure 1.1).

Generally speaking, the term SOLAP refers to the interaction between *static* GIS and *static* data warehouses (i.e., spatial objects and warehouse dimensions that do not change across time). When time gets into play, things become more involved. Different models exist for temporal data warehousing, depending on the approach followed to implement the warehouse [17, 18] . In what follows, we denote a data warehouse *temporal* if it keeps track of the history of the *instances* of the warehouse dimensions, i.e., we assume there are *no structural* (schema) changes.

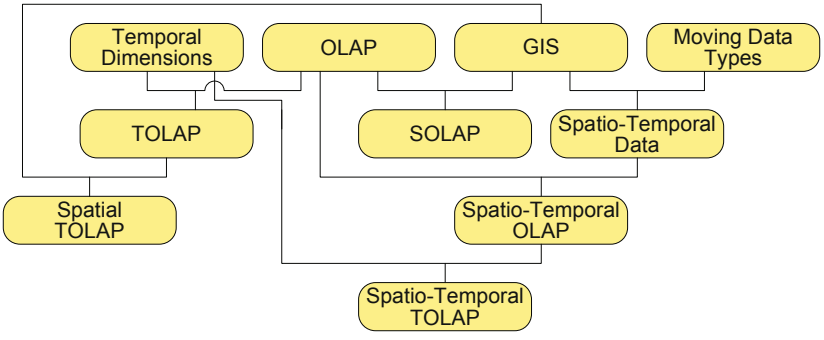


Fig. 1.1. A taxonomy for spatio-temporal data warehousing

In the taxonomy defined in [16] the authors consider four basic classes: Temporal dimensions, OLAP, GIS, and moving data types. Adding moving data types to GIS produces *Spatio-Temporal Data*, typically allowing trajectory analysis in a geographic environment. Providing OLAP with the ability of handling temporal dimensions produces the concept of *Temporal OLAP* (TOLAP). The interaction of OLAP and GIS is denoted *Spatial OLAP* (SOLAP). The interaction between GIS and TOLAP is called *Spatial TOLAP* (S-TOLAP). Adding OLAP capabilities to spatio-temporal data results in *Spatio-Temporal OLAP* (ST-OLAP). Finally, if the latter supports temporal dimensions we have *Spatio-Temporal TOLAP* (ST-TOLAP).

We next discuss the kinds of queries in each one of the classes in the taxonomy of Figure 1.1. The classification is based on the relational calculus with aggregate functions [19]. Then, *OLAP queries* are defined as the ones expressible in such calculus. Let us call $\mathcal{R}_{agg}$ these class of queries. By adding extending $\mathcal{R}_{agg}$ incrementally with new data types, we can analyze the expressive power of each extension. We start by considering a set of non-temporal types (which allow expressing OLAP and SOLAP queries). These include a set of *base types* which are int, real, bool, and string, with the usual

interpretation, *time types* which are instant and periods, and the four *spatial data types*, point, points, line, and region. We have also *Topological predicates* - i.e., touches, which determines whether the boundaries of its two arguments have a nonempty intersection, or overlaps; *set operations*, like crossings, which returns the isolated points in the result of an intersection between two lines, whereas intersection returns common line parts. *Aggregation* operators reduce point sets to points, and *numeric operations*, like no_components, which returns the number of disjoint maximal connected subsets (e.g., faces for a region, connected components for a line graph). There are other spatial functions, like area or distance.

SOLAP queries should be able to compute aggregation of spatial and non-spatial measures over spatial and non-spatial dimensions. Therefore, spatial data types are needed. Then, the class of *SOLAP queries* is the class composed of all the queries that can be expressed by $\mathcal{R}_{agg}$ augmented with spatial data types.

The notion of Temporal OLAP (TOLAP) arises when evolution of the dimension instances in the data warehouse is supported, a problem also referred to as *slowly-changing dimensions* [1]. This evolution is captured by using temporal types. In other words, when at least one of the dimensions in the data warehouse includes a time type, we say that the warehouse supports the TOLAP model. Formally, the class of *TOLAP queries* is the class of queries expressed in $\mathcal{R}_{agg}$ augmented with the time data types.

Spatio-temporal OLAP (ST-OLAP) accounts for the case when the spatial objects evolve over time. For this, the authors consider moving types, denoted moving(α) where α is a spatial data type, and the operations over these data types. For example, the projection of a moving point into the plane may consist of points and lines returned by the operations locations and trajectory, respectively. The projection of a moving line into the plane may consist of lines and regions, returned by the operations routes and traversed. Finally, the projection of a moving region into the plane consists in a region, which is also returned by the operation traversed. Other operations are defined for example, to compute the rate of change for points (e.g., speed, mdirection, etc.). In this way, the class of *ST-OLAP queries* is defined as the language queries expressed by $\mathcal{R}_{agg}$ augmented with spatial types and moving spatial types.

Spatial TOLAP (S-TOLAP) covers the case when in addition to having spatial objects and attributes in the data warehouse, the dimensions are also temporal. Therefore, *Spatial TOLAP queries* are the ones expressible in $\mathcal{R}_{agg}$ augmented with time types, spatial types and moving types.

Finally, Spatio-Temporal TOLAP (ST-TOLAP) is the most general case where there are moving geometries and the dimensions vary over time. That is, *Spatio-Temporal TOLAP* includes the class of queries expressible in $\mathcal{R}_{agg}$ augmented with time types, spatial types, moving spatial types, and moving types.

1.2.2 Conceptual Modeling in Spatio-Temporal OLAP

Several conceptual models have been proposed for spatio-temporal data warehouses. This section is based on the one introduced by Malinowski and Zimányi [20]. The model, called MultiDim, considers a dimension level as spatial if it is represented as a spatial data type (e.g., point, region), and where spatial levels may be related through topological relationships (e.g., contains, overlaps). Spatial measures are characterized as measures representing a geometry, which can be aggregated along the dimensions.

We illustrate our discussion with the following example, where the Environmental Control Agency of a country has a collection of water stations measuring the value of polluting substances at regular time intervals. The application has maps describing rivers, water stations, and the political division of the country into provinces and districts. Figure 1.2 shows the conceptual schema depicting the above scenario using the MultiDim model [20]. There is one fact relationship, WaterPollution, to which several dimensions are related. The fact relationship WaterPollution has two measures, commonArea and load, and is related to five dimensions: Time, Station, Pollutant, River, and District. Dimensions are composed of levels and hierarchies. For example, while the Station dimension has only one level, the District dimension is composed of two levels, District and Province, with a one-to-many parent-child relationship defined between them.

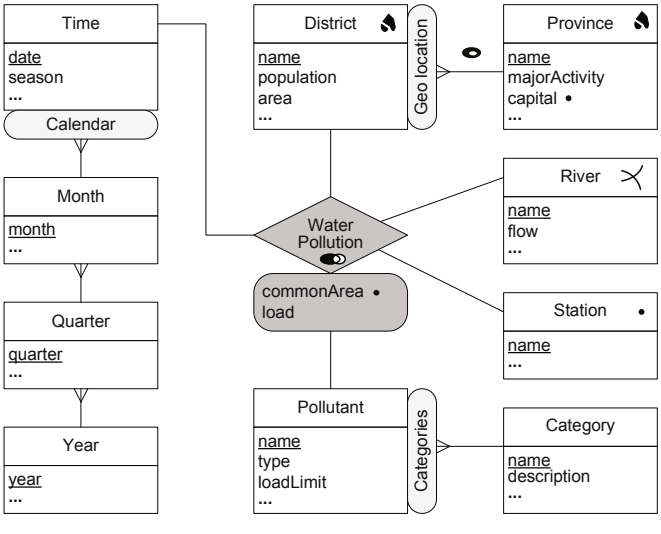


Fig. 1.2. An example of a spatial data warehouse

In the MultiDim model the spatiality of elements is indicated by pictograms. For example, Station, River, and District are spatial levels; they have

a geometry represented by a point, a line, and a region, respectively. Similarly, the attribute capital in Province, as well as the measures commonArea in the three fact relationships are spatial. Finally, topological relationships may be represented in fact relationships and in parent-child relationships. For example, the topological relationship in WaterPollution indicates that whenever a water station, a river, and a district are related in an instance of the relationship, they must overlap. Similarly, the topological relationship in the hierarchy of dimension District indicates that a district is covered by its parent province.

Vaisman and Zimányi [16] extended the Multidim model using the data types defined by Güting *et al.* [21] to the classical base types. They consider *time types* (instant and periods), and four *spatial data types*, point, points, line, and region. Another new data type *Moving* captures the evolution over time of base types and spatial types. Moving types are obtained by applying a constructor moving(·). Hence, a value of type moving(point) is a continuous function f : instant → point. Moving types have associated operations that generalize those of the non-temporal types. This is called *lifting*. For example, a distance function with signature moving(point) × moving(point) → moving(real) calculates the distance between two moving points and gives as result a moving real, i.e., a real-valued function of time. Intuitively, the semantics of such lifted operations is that the result is computed at each time instant using the non-lifted operation.

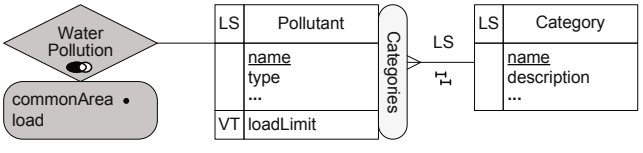


Fig. 1.3. A temporal dimension Pollutant

Temporal dimension levels are identified by the LS pictogram, as shown in Figure 1.3 (taken from [16]). Now, the level Pollutant is temporal, which means that a new pollutant may start to be monitored from a particular date. Temporal levels have a predefined attribute called lifespan, of type moving(bool), which keeps track of the validity of a member at each instant. *Temporal attributes* are identified by the VT pictogram. In the example, the attribute loadLimit is temporal, meaning that the load limit varies across time.

Spatio-temporal objects are represented in the conceptual model as shown in Figure 1.4 (also taken from [16]). The Cloud dimension refers to clouds generated by industrial plants. Both the Cloud level and the commonArea measure have a geometry that is a moving region, indicated by the symbol 'm'. The latter is called a derived measure, i.e., in an instance of the fact relationship that relates a cloud c, a district d, and a date t, the measure

keeps the restriction of the trajectory of the cloud at that date and over that district.

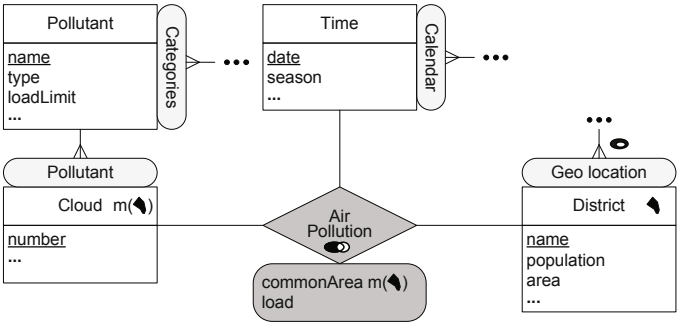


Fig. 1.4. A fact relationship with a spatio-temporal measure and a spatio-temporal dimension

1.2.3 Trajectory Data Warehousing

Moving objects representation and computing have received a fair share of attention over recent years in the database community [22]. The behavior of all these moving objects is traceable by means of electronic devices. We may consider moving objects in two ways: object samples or trajectories. In order to appropriately analyze the latter, some form of interpolation may be necessary. If we assume that moving objects data are captured at a given time interval, with a certain granularity, the *trajectory* of a moving object (MO) is given by samples composed by a finite number of tuples of the form $< Oid, t, x, y >$, stating that at a certain point in time, namely t, the object Oid was located at coordinates (x, y). Trajectory information can be collected and organized in Trajectory Data Warehouses (TDW) in order to be exploited through OLAP and data mining techniques. Applications involving MO analysis include traffic analysis, truck fleet behavior, commuter traffic in a city, passenger traffic in an airport, or shopping behavior in a mall. All of these applications involve aggregation of trajectory data.

The notion of TDW was introduced by Orlando *et al.* [23] . This notion is aimed at providing the infrastructure needed to deliver advanced reporting capabilities and facilitating the use of mining algorithms on aggregated trajectory data. In short, a TDW allows analyzing measures of interest like the number of moving objects in different urban areas, average speed, or speed change. Over the TDW, data mining techniques can also be used to discover traffic-related patterns. An ETL (Extract-Transform-Load) procedure feeds a TDW with aggregate trajectory data, obtained from raw data consisting in the spatio-temporal positions of moving objects. Finally, a data cube is built from the TDW, aggregating measures for OLAP purposes. The trajectories to be analyzed present characteristics of different kinds: numeric (such

as the average speed, direction, duration); spatial (geometric shape of the trajectory), and temporal (timing of the movement).

In order to support trajectory data, a spatio-temporal data cube should allow analysis along (a) temporal dimensions; (b) spatial dimensions at different levels of granularity (point, cell, road); (c) thematic dimensions, containing, for instance, demographic data. In this sense, hierarchies must take into account the fact that an element may rollup to more than one in an upper level. For instance, a road can probably cross more that one cell, yielding a relation instead of a function between a level cell to a level road.

Typically, since the space is usually divided into cells, measures of interest in a TDW are, among other ones: (a) The number of trajectories found in the cell (or started/ended their path in the cell, or crossed/entered/left the cell); (b) The average (or minimum or maximum) distance covered by trajectories in a cell; (c) The average (or minimum or maximum) time required to cover the distance in (b); The speed and change of speed (acceleration), direction and change of direction (turn). Finally, a TDW algebra should support typical OLAP operators like roll-up, drill-down, and slice and dice.

From a modeling point of view, a TDW is based on the classic star schema [1]. It contains a standard temporal dimension, and two spatial dimensions. The former ranges over equally sized time intervals, which are aggregated according to larger intervals as we move up in the dimension hierarchy (e.g., the interval [60,120] aggregates over the interval [0,120]). The spatial dimensions, denoted DimX and DimY, range over equally sized spatial intervals (over the axes x and y, respectively), defining the cells where measures are recorded. There is a fact table containing references to the dimensions and measures of the kinds commented above. Roll-up and drill-down are performed aggregating measures over the cells at different granularities (for instance, combining two or more cells).

Figure 1.5, taken from Damiani et al. [24] depicts the TDW architecture. Initially, location data are captured, and handled by a so-called trajectory stream manager, which builds trajectories from these data (e.g., splitting the raw data according to some criteria), providing a trajectory identifier. This process is called trajectory reconstruction. Trajectories are stored in a relational table, denoted RelTrajectories, and then loaded into a moving object database (MOD). Basically, the MOD includes a relation MODTrajectories with schema (Oid ,trajectoryid, trajectory), where trajectory is of a special type Moving Point.

The ETL process is as follows: Initially, raw location data (usually arriving as a continuous data stream) is transformed into trajectory data. In other words, this step is aimed at determining the starting and ending points of a trajectory. The solution consists in splitting the bulk data according to certain assumptions. For example: (a) Temporal gap (maximum time gap between two points in the same trajectory); (b) Spatial gap (maximum spatial distance between two points); (c) Maximum speed (used to detect noise); (d) Maximum noise duration (if there is a long sequence of noisy observations,

a new trajectory is generated); (e) Tolerance distance D (if two observations are closer than a certain distance D, the latest one is considered redundant).

The TDW can be exploited using OLAP techniques. The aggregated measures allow us to obtain, for example, the variable number of moving objects in different urban areas. Raffaetà et. al. [25] presented T-Warehouse, an implementation that allows analyzing trajectory data at different levels of aggregation.

Marketos and Theodoridis [26] present an extension of the OLAP data model for TDW with the following features: (a) A flexible fact table able to answer queries considering different semantic definitions of trajectories; (b) A parameter that supports the choice of semantics for aggregation queries over trajectory data; (c) An ETL method loading raw location data in the flexible data cube; (d) OLAP techniques to support the different visions explained above. The proposal is denoted ad-hoc OLAP.

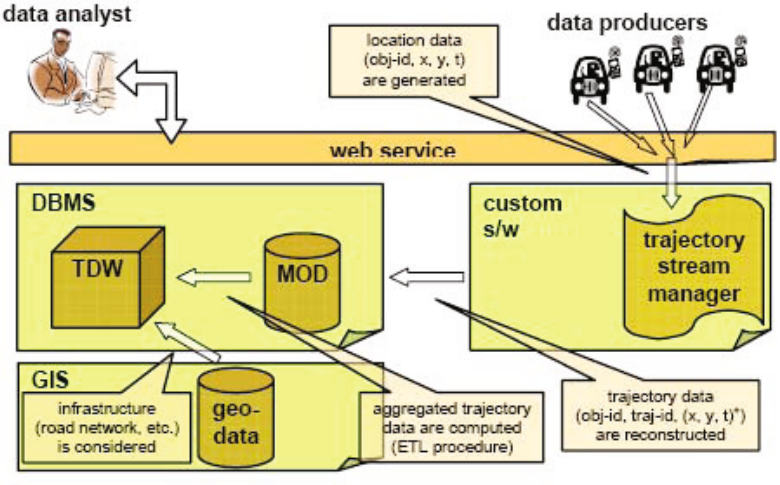


Fig. 1.5. The TDW Process

1.2.4 Next Steps: Handling Raster Data

Advances in data analysis technologies raises new challenges for the BI and GIS research communities. One of them is the need to handle *continuous fields*, which describe physical phenomena that change continuously in time and/or space. Examples of such phenomena are temperature, pressure, and land elevation. Besides physical geography, continuous fields (from now on, fields), like land use and population density, are used in human geography as an aid in spatial decision-making process. Formally, a field is defined as

composed of [27]: (a) a domain $\mathcal{D}$ which is a continuous set; (b) a range of values $\mathcal{R}$; and (c) a mapping function f from $\mathcal{D}$ to $\mathcal{R}$.

In his pioneering work on defining *algebra for fields*, Tomlin [28] proposed a so-called map algebra, based on the notion that a map is used to represent a continuous variable (e.g., temperature). There are three types of functions in Map algebra: *local*, *focal*, and *zonal*. Local functions compute a value at a certain location as a function of the value(s) at this location in other map layer(s). Focal functions compute each location's value as a function of existing values in the neighboring locations of existing layers (i.e., they are characterized by the topological predicate *touches*). Zonal functions (characterized by the topological predicate *inside*), compute a location's new value from one layer (containing the values for a variable), associated to the zone (in another map) containing the location. Câmara *et al.* [29] and Cordeiro *et al.* [30] formalized and extended these functions, supporting more topological predicates. Mennis *et al.* [31] extended map algebra operators for querying time-varying fields.

Although some work has been done to support querying fields in GIS, spatial multidimensional analysis of continuous data is still in its infancy. Existing multidimensional models dealing with discrete data are not adequate for the analysis of continuous phenomena. Multidimensional models and associated query languages are thus needed, to support continuous data. Vaisman and Zimányi [32] presented a conceptual model for SOLAP that supports dimensions and measures representing continuous fields, and characterized multidimensional queries over fields. They defined a *field* data type, a set of associated operations, and a multidimensional calculus supporting this data type. Later on, Gómez at al. [33] proposed an implementation for the operators included in the model.

Figure 1.6, taken from [33], shows how the conceptual model described above was extended to support fields, by adding the notions of field dimensions and field measures. There are pictograms for *temporal* and *non-temporal* field levels and measures. A *field dimension* is a dimension containing at least one level that is a field. In our example, the field dimensions are Elevation, SoilType, Temperature, and Precipitation where the latter two are temporal field dimensions. A *field measure* is a measure represented by a continuous field. For example, suitability is a field measure computed in terms of elements in the model, e.g., the suitability at a certain point can be a function of the kind of crop, temperature, precipitation, and elevation, at that point or its vicinity. Finally, a *field hierarchy* is a set of related field levels; it allows a field to be seen at different granularities.

Notice that in this approach, field dimensions deserve particular treatment. In traditional multidimensional models, every dimension is connected to at least one fact relationship. The same approach has been followed in models introducing fields in spatial data warehouses (e.g., [34]), where dimension instances are values in the underlying domain (that may be obtained through on-the-fly interpolation). Due to the nature of continuous fields, there may be

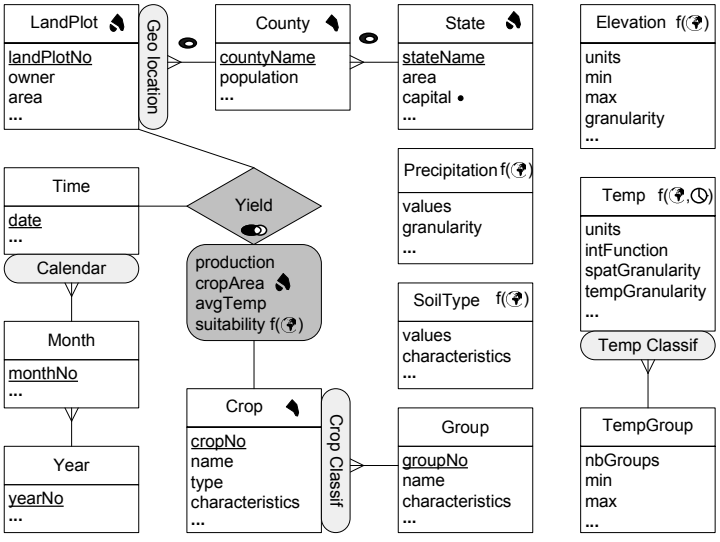


Fig. 1.6. An example of a Spatial data warehouse with continuous fields

an infinite number of instances, each one corresponding to one possible value of the domain. Vaisman and Zimányi [32] follow a different approach: they define a field dimension containing only one instance (corresponding to the function), and the attributes of the field dimension correspond to metadata describing it, like the units at which the values are recorded (e.g., Celsius or Farenheit for temperature). Consequently, field dimensions are part of the model, but are not tied to any particular fact table.

To implement the model, the authors defined two new data types, denoted field and tempfield, and their corresponding operations, again, along the lines of Güting *et al.* [22]. *Field types* capture the variation in space of base types. They are obtained by applying a constructor field(·). Hence, a value of type field(real) (e.g., representing altitude) is a continuous function f : point $\rightarrow$ real. The new type has a set of associated operators. For example, the defspace operator receives a field, and returns the geometry defining it; rangevalues receives a field, and returns the set of values that the function takes. *Aggregation operators* take a field as argument and produce a scalar value. Operations fmin and fmax give, respectively, the minimum and maximum value taken by the function. The Map Algebra operators are also supported.

There is still much work to do in this topic. In real-world practice, scientists and practitioners register the values of a field taking samples at (generally) fixed locations, and inferring the values at other points in space using some interpolation method. The raster model is just one of the discrete data models proposed to represent fields based on sampling and interpolation. The Map Algebras commented above, have been proposed to work over raster

representations, although Triangular Irregular Networks (TIN) [35] and Voronoi diagrams [36] are becoming increasingly popular. Moreover, the arguments of the operators must be fields of the same kind. One research direction therefore can be to define a general algebra for spatio-temporal multidimensional data over continuous fields, independently of the underlying representation of such data. These algebras should easily support new representations that may appear in the future.

1.3 Real-Time Data Warehousing

1.3.1 The Need for Real-Time Data Warehousing

The typical method of updating data in a DW (or Data Mart) consists in implementing the well-known extraction, transformation and load (ETL) process (discussed above in the framework of trajectory DW). In a nutshell, ETL tools pull data from source systems periodically e.g., daily, weekly, monthly), providing a snapshot of the business data at a given moment in time. These batch data are then loaded into the data warehouse tables. During each cycle, the warehouse tables are refreshed and the process is repeated (in general, no matter whether the data has changed or not). Historically, this process has been considered acceptable, since it was next to impossible to get Real-Time (RT), continuous DW feeds from production systems. Moreover, it was difficult to get consistent, reliable results from query analysis if warehouse data was constantly changing. Nowadays, however, DW users want current and up-to-date BI information. In addition, while in the early days only selected users accessed most DWs, in today's web-based architectures large volumes of concurrent requests must be handled maintaining consistent query response times, and must scale seamlessly as the data volume and number of users grows continuously. Moreover, DW need to remain available 7 x 24. To be more technical, let us comment on the lifecycle of a data record in a BI environment. The cycle starts with a business event taking place. ETL routines then deliver the event record to the DW. Finally, analytical processing turns the data into information, to help the decision-making process. and a business decision leads to a corresponding action. To approach Real Time, the time elapsed between the event and its consequent action (called the *data latency*) needs to be minimized. In the general case, it is the data acquisition process that introduces majority of the data latency. Therefore, to support real-time BI, real-time DW are needed (RTDW) [37]. Examples of these needs were discussed by Schneider [38], and include for instance: (a) Collaborative filtering, e.g., with queries such as "People who like X also like Y". Here, the timeliness (freshness of data) is in the range of hours; (b) Fraud Detection. Detects anomalies in credit card usage. Timeliness here is in the order of minutes; (c) Call Center applications, e.g., to provide next best offer or action. In this applications, timeliness is again, minutes. (d) Web Page Usage analysis, e.g., by property, geography, user demographics, referrer, etc. Timeliness in this

case in in the range of hours or one day at most. (e) Business Activity Monitoring and Operational Performance Management (e.g., real-time inventory analysis). Timeliness would be in the order of minutes.

Making rapid decisions based on large volumes of data requires achieving low data latency, sometimes at the expense of potential data inconsistency (e.g., late, missing data) and *very* high availability requirements, and specialized hardware. Further, integration with other sources can be a challenge. We next discuss how the DW community has addressed these challenges. However, the reader must be aware of the fact that, in spite that many applications require very low latency access, on the other hand, most applications do still not require these latency levels (that may come down to the seconds granularity). For example, some applications like targeting, alerting, recommendations, may demand for no so fresh data. In these cases, the common evolution strategy is to increase frequency of ETL operations using mini-batch ETL, e.g., loading data every 10 minutes.

1.3.2 Strategies for Enabling Real Time ETL

Several alternatives have been devised to achieve RT ETL in order to reduce data latency. The simplest one, which requires the least effort in terms of changes to existing architectures, is the one called Near-Real Time ETL. Near RT ETL tools simply increase the frequency of ETL without changing the load process, data models and reporting applications. Most work in the field follows this approach (see e.g. [39]). However, this is not enough when data latency must be drastically reduced, and other alternative paths must be followed. These paths take advantage of the fact that data warehouses become more 'operational' in a RT data scenario, and transformations occur at the data warehouse, thus reducing data and analysis latency, eliminating the need to aggregate updated data on a centralized server until it is batch-processed. The former led to alternatives for approaches like the *Direct Trickle Feed* (DTF). Here, new data from the operational sources are continuously fed into the data warehouse This is done by either directly inserting data in the fact tables, or by inserting data into separate fact tables in a dedicated real-time partition. A variant of this strategy, which addresses the mixed workload problem introduced by DTF, is the one called *Trickle and Flip*. In this approach, instead of loading the data in real-time into the actual warehouse tables, data are continuously fed into staging tables that are an exact copy of the target tables. Periodically, feeding is stopped, and the copy is swapped with the fact table, bringing the DW up-to-date. Although cycle times range from hours to minutes, the best reported performances range between five and ten minutes. Finally, the *Real-Time Data Caching* (RTDC) strategy [40], although costly, completely avoids mixed workload problems: a RTDC consists in a dedicated database server (or a separate instance of a large database system) dedicated to loading, storing, and processing the RT data. In-Memory databases could be used for RTDC for large volumes of

RT data (in the order of hundreds or thousands of changes per second), or extremely fast query performance requirements Here, the RT data is loaded into the cache as it arrives from the source system. A drawback of this strategy is that, since the RT and historical data are separately stored, when a query involves both kinds of data, its evaluation could be costly.

Modeling RT Fact Tables

Physical modeling help the strategies above in reducing data latency. A classic solution consists in define Real-Time Fact Table partitions, where RT and historical data are stored in separate fact tables [41]. This partition is physically and administratively separated from the Data Warehouse tables, and it is subject to special update and query rules. The RT partition must resemble as much as possible the static data warehouse fact tables, and must contain all the activity that has occurred since the last update of the static data warehouse. Query tools should be able to distinguish both kinds of tables, and know where to find data. That means, the BI tools must be smart enough to formulate a query to drill across the static fact tables, and include the up-to-the-second records from the real-time partition. This is not always achieved by commercial BI tools, however.

An alternative to this solution consists in modeling with an external Real-Time data cache. In this case, no new modeling is required, and the RDTC has the same structure as historical data.

1.3.3 ETL or CFT?

In a scenario where data refreshment tools must provide real-time access, the larger the data warehouse, the longer it takes to refresh with the traditional ETL procedure. More than often, the volume of data being loaded into the warehouse exceeds the allocated batch updating window. Then, in an environment where 20 per cent of operational data change every week or month, some practitioners and vendors propose a Capture, Transform and Flow (CTF) solution that captures, transforms and flows data in real-time into an efficient, continuously refreshed data warehouse [42]. We discuss this next.

Data Capture

With advanced CTF solutions, every time an add, change or delete occurs in the production environment, it is automatically captured and integrated or pushed in real-time to the data warehouse. Beyond real-time integration, change data capture can also be done periodically. Data can be captured and then stored until a predetermined integration time. For example, an organization may schedule its refreshes of full tables or changes to tables to be integrated hourly or nightly. Only data that has changed since the previous integration needs to be transformed and transported to the subscriber.

The data warehouse can therefore be kept current and consistent with the source databases. Figure 1.7 depicts the procedure followed by the Oracle Data Integration Tool[2]. First, an identified subscriber (e.g., an integration process) subscribes to changes that might occur in a datastore. The Changed Data Capture framework then captures changes in the datastore and publishes them for the subscriber. Finally, the subscriber can process the tracked changes at any time and consume these events. Once consumed, events are no longer available for this subscriber.

Data Transformation

Transformational data integration software can conduct individual tasks such as translating values, deriving new calculated fields, joining tables at source, converting date fields, and reformatting field sizes, table names and data types. All of these functions allow for code conversion, removal of ambiguity and confusion associated with data, standardization, measurement conversions, and consolidating dissimilar data structures for data consistency.

Data Flow

This refers to refreshing the feed of transformed data in real-time from multiple operational systems to one or more subscriber systems. The flow process is a continuous stream of information as opposed to the batch loading of data performed by ETL tools.

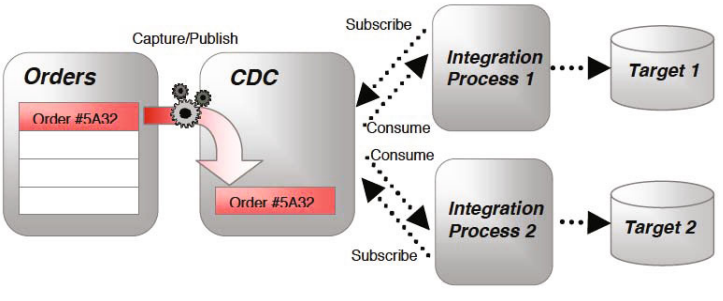


Fig. 1.7. Change Data Capture

1.3.4 Right Time Data Warehousing

Thomsen et al. [43] proposed the RiTE architecture for what they denoted *Right Time DW* (see Figure 1.8). Here, a so-called RiTE 'Catalyst' module is added to the classical ETL architecture. The rationale is based on the fact

[2] http://www.oracle.com

that some data must be fresh while other data can support higher latency, that is, parts of the data must be loaded quickly after arrival, while other parts can be loaded at regular intervals In other words, the data is loaded at the *right time*, therefore the approach is called "right-time data warehousing", opposite to "near-real time DW", were data is loaded into the DW minutes or seconds after it arrives. In both approaches, regular SQL INSERT statements are used, leading to slow insert speed. An appropriate solution should find the correct batch size between the two extremes (bulk load vs. single-row INSERT). The RiTE architecture includes:

- A specialized JDBC database driver for the producer
- A specialized JDBC database driver for the consumers
- A main-memory based catalyst which provides intermediate storage (memory tables) for (user-chosen) DW tables, offers fast insertions and concurrency control. Besides, data can be queried while held by memory tables, transparently to the end user. Eventually the data is moved to its final target the physical DW tables
- A PostgreSQLtable function makes the data available in the DW

The data producer uses a specialized database driver which handles INSERT and COMMIT operations in a particular way. Other operations are executed traditionally. The RiTE prototype treats prepared statements inserting scalars into memory tables, and can handle typical DW fact tables The values are not inserted directly into the DW, but instead kept in a local buffer. Later the data is flushed and reaches the memory table Finally, the data is materialized and reaches the DW tables. The default is to flush immediately after the commit although another policy waits and temporarily places the data at a file on the producer side (this is called the "lazy commit").

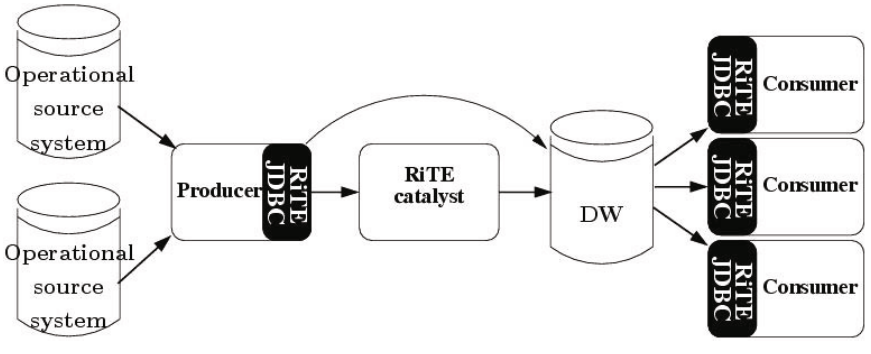


Fig. 1.8. The RiTE Architecture

1.3.5 Future Steps

In spite of the work commented here, we believe that there is a lot of room for fruitful research here. Most of the work has been done in the ETL process, mostly at the physical level. However, less work has analyzed whether or not the current data models are still appropriate when data latency is a hard requirement. Therefore, comprehensive models and frameworks are needed to allow RTDW on solid theoretical basis. Re-reading the needs stated in [44], we can see that this area has evolved, from the research and scientific point of view, more slowly than other areas (like Spatial OLAP and BI), were solid scientific results were achieved.

1.4 Semantic Web Data Warehouses

The *Semantic Web* (SW) is a proposal oriented to represent Web content in an easily machine-processable way. The basic layer of the data representation for the Semantic Web recommended by the World Wide Web Consortium (W3C) is the Resource Description Framework (RDF) [11]. The Ontology Web Language (OWL)[3] is a language for the specification of ontologies, whose definition by the W3C Consortium has encouraged different communities to develop large and complex ontologies like the NCI thesaurus, GALEN, etc. OWL provides a powerful knowledge representation language that has a clean and well defined semantics based on Description Logics (DL). In a Semantic Web scenario, domain ontologies (defined in RDF or some variant of OWL) are aimed at defining a common terminology for the concepts involved in a particular domain. Semantic annotations are especially useful for describing unstructured, semi-structured and text data. Many applications attach metadata and semantic annotations to the information they produce (for example, in medical applications, medical image, laboratory tests). In the near future, large repositories of semantically annotated data will be available, opening new opportunities for enhancing current decision support systems. Perez et al. [45] provide a good survey of the efforts made for integrating web data analysis into OLAP tools and techniques, although oriented in a different direction than the present discussion: the authors focus on XML data warehouses and tools, while we are aimed at discussing pure semantic web data analysis. Two lines of work are clearly identified. One focuses in multidimensional design automation taking advantage of existing ontologies. The other one studies methods aimed at analyzing the large amounts of semantic web data using OLAP tools. We believe the latter is a very promising line of work.

1.4.1 DW Design Automation from Ontologies

There is a large corpus of work aimed at using ontologies for automatic DW design. As an example, Niinimäki and Niemi [46] use semantic web technologies to populate OLAP cubes. They use ontology mapping to convert data

[3] http://www.w3.org/TR/owl2-overview/

sources to RDF and then query this RDF data with SPARQL3 to populate the OLAP schema. The method uses an ontology and mapping files that connect the data sources with the ontology. The ETL process is guided by the ontology. The authors create an OLAP ontology (e.g., defining that there are dimensions and measures, and that the measure is a function of the dimensions). From this ontology they build application specific sub-ontologies. Data are then formatted to conform to these ontologies. Ontologies are expressed in RDF and OWL. Along the same lines, Romero and Abelló [47] address the design of the data warehouse starting from an OWL ontology that describes the data sources. They identify the dimensions that characterize a central concept under analysis (the fact concept) by looking for concepts connected to it through one-to-many relationships. The same idea is used for discovering the different levels of the dimension hierarchies, starting from the concept that represents the base level. In this work the input ontology indicates the multiplicity of each role in the relationships; and a matrix keeps, for each concept, all the concepts that are related by means of a series of one-to-many relationships. The output of the Romero and Abelló's method is a star or snowflake schema that guaranties the summarizability of the data, suitable to be instantiated in a traditional multidimensional database. The application of this work is valid in scenarios where a single ontology of reduced size, with multiplicity restrictions, is used for annotating the source data. Note that multiplicity information is rarely found in the source ontologies. Schematically, the method is composed of three clearly defined steps, each one aimed at discovering an specific multidimensional concept automatically. At the end of each step the method asks for the end-user multidimensional requirements (the only non-automatic part of the process). The first task looks for potential facts. Therefore, concepts with most potential dimensions and measures are good candidates. At the end of this task, the user is required to choose the subjects of interest among the concepts proposed as potential Facts. The second task points out sets of concepts likely to be used as a base for each identified fact. A base is defined as a collection of concepts labeled as potential dimensions. That is, the method now looks for concepts being able to univocally identify objects of analysis (i.e., factual data). The third task produces dimension hierarchies. For every concept identified as a dimension, a hierarchy is built from those concepts related to each other by typical whole-part relationships.

1.4.2 Analyzing Ontology Instances

Following the second line of research that we mentioned above, Nebot el al. [48] proposed a semi-automatic method for on-demand extracting semantic data into a multidimensional database. In this way, data could be analyzed using traditional OLAP techniques. In other words, an ETL process for Semantic Web data sources. Figure 1.9, taken from the mentioned work, shows

a very general sketch of how ontology data are transformed into multidimensional data. The authors present a method for identifying, transforming and combining semantic data in order to discover facts and populate a MD model with such facts. Figure 1.10 depicts the general architecture of the proposal. The authors assume that data is represented as an OWL ontology. Here, data are separated into the TBox (the 'schema'), and the ABox (the 'instance'). The former is usually called the 'ontology', while the latter are the "semantic annotations', represented as triples where the subject is always an instance, an object can be either an instance or a literal, and the predicate can be either an object property or a data type property. Figure 1.10 shows the ontology and the annotations, from which data to populate the fact table are indicated. There are three main phases in the methodology: (1) Designing a MD schema. The user selects the subject of analysis, which corresponds to a concept of the ontology. Then, she selects potential dimensions, either a named concept or data type property. Finally, she defines the measures, which are functions over data type properties. (2) The fact extractor identifies and extracts facts from the instance store according to the MD schema previously designed, producing the base fact table of a data warehouse. (3) The user specifies MD queries over the data warehouse. The MD query is executed and a cube is built. Then, typical OLAP operations can be applied over the cube.

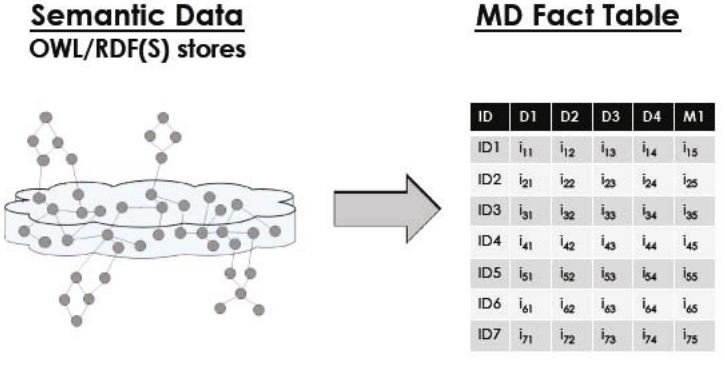


Fig. 1.9. From Ontologies to Multidimensional Data

1.4.3 Scaling OLAP for the Semantic Web Analysis

We have mentioned previous efforts on analyzing semantic web data using OLAP tools. In a semantic web context, where data are represented as an RDF labeled graph (i.e., binary relations), analytical queries consist of three main constructs namely graph pattern matching, grouping and aggregation.

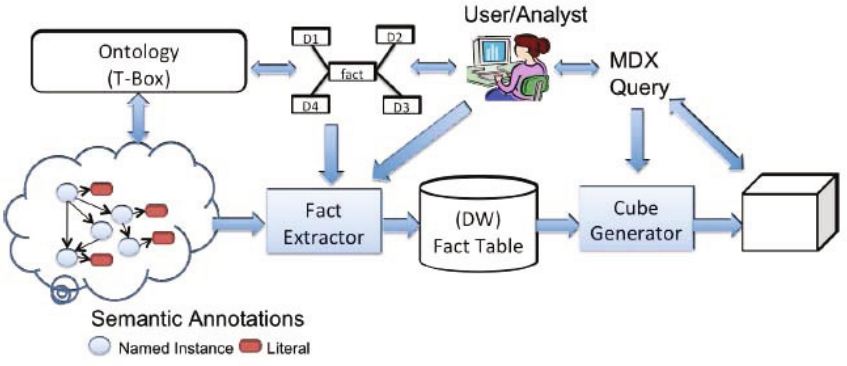


Fig. 1.10. Architecture

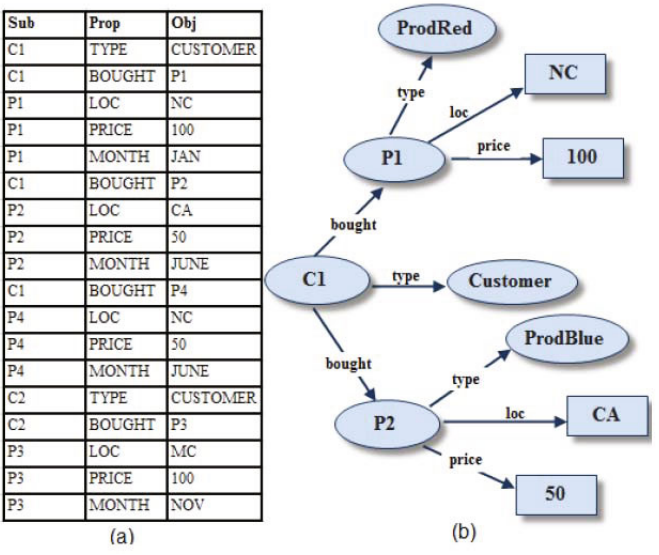
Therefore, join operations are needed to transform data into *n-ary* relations relevant to the given query. On the contrary, in traditional OLAP, data are organized in a structured fashion. Processing semantic web data implies that powerful processing mechanisms are needed, given that large joins must be computed. Thus, parallel processing systems are needed. We commented in Section 1.1 that implementations of Google's MapReduce [49] are gaining success in processing large analytical workloads, although the model is not appropriate to compute costly joins.

Remark 1. As a solution to the problem of join computing, a new hybrid approach, HadoopDB [50], is becoming increasingly popular. It combines the performance and efficiency advantages of parallel databases with the scalability and fault-tolerance advantages of MapReduce systems. Note however that HadoopDB partitions the data as required by the query and pushes as many operations as possible to the database, which reduces the improvement to the processing of the first reduce task, which usually marks the end of the first join operation. This is not enough for RDF processing. Therefore, Afrati and Ullman [51] optimize multi-way joins by providing strategies to efficiently partition and replication of the tuples of a relation on reducer processes, minimizing the communication cost. Other query optimization techniques used in RDF query processing, consist in materializing the transitive closure of the RDF graph. □

In the MapReduce programming model, tasks are encoded using two independent functions: Map and Reduce, such that their execution can be parallelized on a cluster of machines. The Map function reads each line in the data and transforms it into a $\langle key, value \rangle$ pair. After all mapping tasks are completed, the intermediate keys are sorted and merged. The objects with the same

key i.e. $\langle key, list(values)\rangle$ are collected by an specific reducer. The Reduce function focuses on performing aggregation operations on this grouped data. Sometimes a combiner function may be used to perform partial aggregations in the map phase. However, this model was devised with a single large data file in mind, not in a Semantic Web scenario, where matching graph patterns results in a large number of joins. Graph pattern matching is the typical query evaluation mechanism of SPARQL, the standard query language for the Semantic Web[4].

To enhance parallelism of analytical processing on RDF graphs, Sridhar et al. [52] proposed a dataflow language, denoted RAPID, which extends Yahoo's Pig Latin language with query primitives for dealing with the graph structured nature of RDF. They also propose a technique to express complex analytical queries in a way that allows optimization and enhances parallelization on MapReduce platforms. This technique integrates into Pig Latin primitives for implementing the MD-join operator [53]. As an example, Figure 1.11 taken from [52] shows a very simple fact table, expressed as an RDF graph, that can lead to difficulties to evaluate aggregate queries like "Number of sales, for each product and month of 2000, such that sales were between the previous and following months' average sales".



Sub	Prop	Obj
C1	TYPE	CUSTOMER
C1	BOUGHT	P1
P1	LOC	NC
P1	PRICE	100
P1	MONTH	JAN
C1	BOUGHT	P2
P2	LOC	CA
P2	PRICE	50
P2	MONTH	JUNE
C1	BOUGHT	P4
P4	LOC	NC
P4	PRICE	50
P4	MONTH	JUNE
C2	TYPE	CUSTOMER
C2	BOUGHT	P3
P3	LOC	MC
P3	PRICE	100
P3	MONTH	NOV

Fig. 1.11. A simple fact table in RDF

Ravindra et al. [54] extend the previously commented work developing User Defined Functions (UDFs) , through basically two mechanisms: function

[4] http://www.w3.org/TR/rdf-sparql-query/

coalescing (i.e., merging MapReduce functions into a single one, to reduce function calls and I/O) and look-ahead processing. These UDFs have been integrated into the *Pig Latin* function library and the experimental results, according to the authors, show up to a 50% improvement in execution times for certain classes of queries.

1.5 Conclusion

We have discussed new directions in DW and BI, addressing quite consolidated fields, like spatio-temporal BI, topics that in spite of having been detected as critical (like real-time DW) have not produced relevant results so far, and topics that we expect will receive attention in the near future, like Semantic Web DW. From our discussion, it follows that future research will focus on scalability and performance, given the increasing amount of available and heterogeneous data. More formal work on Real-Time DW, particularly in modeling is needed, as well as models and methodologies for DW and BI on the Semantic Web. In the latter case, proposals so far are ad-hoc and relying on transforming RDF graphs and ontologies into standard Star-schemas. Finally, new domains for BI are slowly emerging, like image and multimedia OLAP [55], and will also require new models and algorithms to be developed.

References

1. Kimball, R.: The Data Warehouse Toolkit. J. Wiley and Sons (1996)
2. Cabibbo, L., Torlone, R.: Querying Multidimensional Databases. In: Cluet, S., Hull, R. (eds.) DBPL 1997. LNCS, vol. 1369, pp. 253–269. Springer, Heidelberg (1998)
3. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing data cubes efficiently. In: Proceedings of SIGMOD, Montreal, Canada, pp. 205–216 (1996)
4. Stonebraker, M.: Stonebraker on data warehouses. Commun. ACM 54(5), 10–11 (2011)
5. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Commun. ACM 53(1), 72–77 (2010)
6. Stonebraker, M., Abadi, D.J., DeWitt, D.J., Madden, S., Paulson, E., Pavlo, A., Rasin, A.: MapReduce and parallel DBMSs: friends or foes? Commun. ACM 53(1), 64–71 (2010)
7. Bajda-Pawlikowski, K., Abadi, D.J., Silberschatz, A., Paulson, E.: Efficient processing of data warehousing queries in a split execution environment. In: Proceedings of SIGMOD, pp. 1165–1176 (2011)
8. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, N.Z.S., Liu, H., Murthy, R.: Hive: a petabyte scale data warehouse using Hadoop. In: Proceedings of ICDE, pp. 996–1005 (2010)
9. Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sarma, J.S., Murthy, R., Liu, H.: Data warehousing and analytics infrastructure at facebook. In: Proceedings of SIGMOD, pp. 1013–1020 (2010)
10. Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: Mad skills: New analysis practices for big data. PVLDB 2(2), 1481–1492 (2009)

11. Lassila, O., Swick, R.R. (eds.): Resource description framework (RDF) model and syntax specification. W3C Recommendation (1999)
12. Worboys, M.F.: GIS: A Computing Perspective. Taylor & Francis (1995)
13. Rivest, S., Bédard, Y., Marchand, P.: Toward better support for spatial decision making: Defining the characteristics of spatial on-line analytical processing (SOLAP). Geomatica 55(4), 539–555 (2001)
14. Shekhar, S., Lu, C., Tan, X., Chawla, S., Vatsavai, R.R.: Mapcube: A visualization tool for spatial data warehouses. In: Miller, H.J., Han, J. (eds.) Geographic Data Mining and Knowledge Discovery, pp. 74–109. Taylor & Francis (2001)
15. Gómez, L., Haesevoets, S., Kuijpers, B., Vaisman, A.A.: Spatial aggregation: Data model and implementation. CoRR abs/0707.4304 (2007)
16. Vaisman, A., Zimányi, E.: What is Spatio-Temporal Data Warehousing? In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 9–23. Springer, Heidelberg (2009)
17. Eder, J., Koncilia, C., Morzy, T.: The COMET Metamodel for Temporal Data Warehouses. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 83–99. Springer, Heidelberg (2002)
18. Mendelzon, A.O., Vaisman, A.A.: Temporal queries in OLAP. In: Proceedings of VLDB, Cairo, Egypt, pp. 242–253 (2000)
19. Klug, A.: Equivalence of relational algebra and relational calculus query languages having aggregate functions. Journal of ACM (1982) 699–717
20. Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications. Springer, Heidelberg (2008)
21. Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., Vazirgiannis, M.: A foundation for representing and quering moving objects. ACM Trans. Database Syst. 25(1), 1–42 (2000)
22. Güting, R.H., Schneider, M.: Moving Objects Databases. Morgan Kaufmann (2005)
23. Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., Silvestri, C.: Spatio-Temporal Aggregations in Trajectory Data Warehouses. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 66–77. Springer, Heidelberg (2007)
24. Damiani, M.L., Vangenot, C., Frentzos, E., Marketos, G., Theodoridis, Y., Veryklos, V., Raffaetà, A.: Design of the trajectory warehouse architecture. Technical Report D1.3, GeoPKDD project (2007)
25. Raffaetà, A., Leonardi, L., Marketos, G., Andrienko, G.L., Andrienko, N.V., Frentzos, E., Giatrakos, N., Orlando, S., Pelekis, N., Roncato, A., Silvestri, C.: Visual mobility analysis using t-warehouse. International Journal of Data Warehouse and Mining 7(1), 1–23 (2011)
26. Marketos, G., Theodoridis, Y.: Ad-hoc OLAP on trajectory data. In: Proceedings of MDM, pp. 189–198 (2010)
27. Paolino, L., Tortora, G., Sebillo, M., Vitiello, G., Laurini, R.: Phenomena: a visual query language for continuous fields. In: Proceedings of ACM-GIS, pp. 147–153 (2003)
28. Tomlin, C.D.: Geographic Information Systems and Cartographic Modelling. Prentice-Hall (1990)
29. Câmara, G., Palomo, D., de Souza, R.C.M., de Oliveira, O.R.F.: Towards a generalized map algebra: Principles and data types. In: Proceedings of GeoInfo., pp. 66–81 (2005)

30. Cordeiro, J.P., Câmara, G., Moura, U.F., Barbosa, C.C., Almeida, F.: Algebraic formalism over maps. In: Proceedings of GeoInfo., pp. 49–65 (2005)
31. Mennis, J., Viger, R., Tomlin, C.D.: Cubic map algebra functions for spatiotemporal analysis. Cartography and Geographic Information Science 32(1), 17–32 (2005)
32. Vaisman, A.A., Zimányi, E.: A multidimensional model representing continuous fields in spatial data warehouses. In: Proceedings of ACM-GIS, pp. 168–177 (2009)
33. Gómez, L., Vaisman, A., Zimányi, E.: Physical Design And Implementation of Spatial Data Warehouses Supporting Continuous Fields. In: Bach Pedersen, T., Mohania, M.K., Tjoa, A.M. (eds.) DAWAK 2010. LNCS, vol. 6263, pp. 25–39. Springer, Heidelberg (2010)
34. Ahmed, T.O., Miquel, M.: Multidimensional Structures Dedicated to Continuous Spatiotemporal Phenomena. In: Jackson, M., Nelson, D., Stirk, S. (eds.) BNCOD 2005. LNCS, vol. 3567, pp. 29–40. Springer, Heidelberg (2005)
35. Kumler, M.P.: An intensive comparison of triangulated irregular networks (TINs) and digital elevation models (DEMs). Cartographica 31(2), 1–99 (1994)
36. Ledoux, H., Gold, C.: A Voronoi-based map algebra. In: Riedl, A., Kainz, W., Elmes, G.A. (eds.) Progress in Spatial Data Handling, pp. 117–131. Springer, Heidelberg (2006)
37. Bruckner, R.M., List, B., Schiefer, J.: Striving Towards Near Real-Time Data Integration for Data Warehouses. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2002. LNCS, vol. 2454, pp. 317–326. Springer, Heidelberg (2002)
38. Schneider, D.A.: Practical Considerations for Real-Time Business Intelligence. In: Bussler, C.J., Castellanos, M., Dayal, U., Navathe, S. (eds.) BIRTE 2006. LNCS, vol. 4365, pp. 1–3. Springer, Heidelberg (2007)
39. Simitsis, A., Vassiliadis, P., Sellis, T.K.: Optimizing ETL processes in data warehouses. In: Proceedings of ICDE, pp. 564–575 (2005)
40. Zhu, Y., An, L., Liu, S.: Data updating and query in real-time data warehouse system. In: Proceedings of CSSE, pp. 1295–1297. IEEE Computer Society, Washington, DC, USA (2008)
41. Kimball, R., Ross, M.: The Kimball Group Reader: Relentlessly Practical Tools for Data Warehouse and Business Intelligence. J. Wiley and Sons (2010)
42. Vandemay, J.: Considerations for building a real-time data warehouse. Technical Report DMC (White Paper), Data Mirror Corporation (2001)
43. Thomsen, C.S., Pedersen, T.B., Lehner, W.: RiTE: Providing on-demand data for right-time data warehousing. In: Proceedings of ICDE, pp. 456–465. IEEE Computer Society, Washington, DC, USA (2008)
44. Hammer, J., Schneider, M., Sellis, T.: Data warehousing at the crossroads. Technical Report 04321, Dagsthul Seminar (2004)
45. Pérez, J.M., Llavori, R.B., Aramburu, M.J., Pedersen, T.B.: Integrating data warehouses with web data: A survey. IEEE Trans. Knowl. Data Eng. 20(7), 940–955 (2008)
46. Niinimäki, M., Niemi, T.: An ETL process for OLAP using RDF/OWL ontologies. Journal on Data Semantics 13, 97–119 (2009)
47. Romero, O., Abelló, A.: Automating multidimensional design from ontologies. In: Proceedings of DOLAP, pp. 1–8 (2007)
48. Nebot, V., Llavori, R.B.: Building data warehouses with semantic data. In: Proceedings of EDBT/ICDT Workshops (2010)

49. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)
50. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D.J., Rasin, A., Silberschatz, A.: HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. PVLDB 2(1), 922–933 (2009)
51. Afrati, F.N., Ullman, J.D.: Optimizing joins in a map-reduce environment. In: Proceedings of EDBT, pp. 99–110 (2010)
52. Sridhar, R., Ravindra, P., Anyanwu, K.: RAPID: Enabling Scalable Ad-Hoc Analytics on the Semantic Web. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 715–730. Springer, Heidelberg (2009)
53. Chatziantoniou, D., Akinde, M.O., Johnson, T., Kim, S.: The MD-join: An operator for complex OLAP. In: Proceedings of ICDE, pp. 524–533 (2001)
54. Ravindra, P., Deshpande, V.V., Anyanwu, K.: Towards scalable RDF graph analytics on MapReduce. In: Proceedings of MDAC, vol. 5, pp. 1–5 (2010)
55. Jin, X., Han, J., Cao, L., Luo, J., Ding, B., Lin, C.X.: Visual cube and on-line analytical processing of images. In: Proceedings of CIKM, pp. 849–858 (2010)

2

Data Warehouse Performance: Selected Techniques and Data Structures

Robert Wrembel

Institute of Computing Science
Poznań University of Technology
Poznań, Poland
Robert.Wrembel@cs.put.poznan.pl

Summary. Data stored in a data warehouse (DW) are retrieved and analyzed by complex analytical applications, often expressed by means of star queries. Such queries often scan huge volumes of data and are computationally complex. For this reason, an acceptable (or good) DW performance is one of the important features that must be guaranteed for DW users. Good DW performance can be achieved in multiple components of a DW architecture, starting from hardware (e.g., parallel processing on multiple nodes, fast disks, huge main memory, fast multi-core processor), through physical storage schemes (e.g., row storage, column storage, multidimensional store, data and index compression algorithms), state of the art techniques of query optimization (e.g., cost models and size estimation techniques, parallel query optimization and execution, join algorithms), and additional data structures improving data searching efficiency (e.g., indexes, materialized views, clusters, partitions). In this chapter we aim at presenting only a narrow aspect of the aforementioned technologies. We discuss three types of data structures, namely indexes (bitmap, join, and bitmap join), materialized views, and partitioned tables. We show how they are being applied in the process of executing star queries in three commercial database/data warehouse management systems, i.e., Oracle, DB2, and SQL Server.

Keywords: data warehouse, star query, join index, bitmap index, bitmap join index, materialized view, query rewriting, data partitioning, Oracle, SQL Server, DB2.

2.1 Introduction

A *data warehouse architecture* has been developed in order to integrate and analyze data coming from multiple distributed, heterogeneous, and autonomous data sources (DSs), deployed throughout an enterprise. A core component of this architecture is a database, called a *data warehouse* (DW), that stores current and historical data, integrated from multiple DSs. The content of a DW is analyzed by various On-Line Analytical Processing (OLAP) applications for the purpose of discovering trends (e.g., demand and sales of products), discovering patterns of behavior (e.g., customer habits, credit repayment history) and anomalies (e.g., credit card usage) as well as for finding

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 27–62, 2012.

dependencies between data (e.g., market basket analysis, suggested buying, insurance fee assessment). OLAP applications execute complex queries, some of them being predefined (e.g. reports), whereas others being executed ad-hoc by decision makers.

The queries are expressed with multiple join, filtering, aggregate, and sort operations and they process large (or extremely large) volumes of data. A special class of analytical queries includes *star queries* that join a central table with multiple referenced tables. The execution of analytical queries may take hours or even days. For this reason, providing means for increasing the performance of a data warehouse for analytical queries and other types of data processing (e.g., data mining) is one of the important research and technological areas.

2.1.1 Increasing DW Performance: Research and Technological Advances

A DW performance depends on multiple components of a DW architecture. The basic components include: (1) hardware on which a database/data warehouse management system (DB/DWMS) is installed and computational architectures, (2) physical storage schemes of data, (3) robustness of a query optimizer, (4) additional data structures supporting faster data access.

Hardware Architectures

Hardware and various processing architectures, e.g., shared memory, shared disk, shared nothing, have a substantial impact on the performance of a DW. Multiple nodes with their processing power allow to process data in parallel. Modern cluster, grid, and cloud architectures take advantage of parallel data access and processing. A lot of research efforts focus on this area, e.g., [1, 2, 3]. Recent research and technological trends concentrate also on using parallel processing of powerful graphic processing units, e.g,. [4, 5, 6, 7, 8] for processing data. Another hot topic research and technological issue concerns main memory databases and data warehouses, e.g., [9, 10, 11].

Physical Storage

In practice, a DW is implemented either in a relational server (the ROLAP implementation) or in a multidimensional server (the MOLAP implementation). The ROLAP implementation can be based either on a row storage (RS) (a typical one) or on a column storage (CS) (current trend). RS is more suitable for transactional processing (inserts, deletes, updates). CS offers better performance for applications that read and compute aggregates based on the subset of table columns, thus it is better suited for OLAP processing. An intensive research is conducted in the area of column store DWMSs, e.g.,

[12, 13, 14] in order to develop efficient join algorithms, query materialization techniques, index data structures, and compression techniques [15], to mention some of them. These and many other efforts resulted in commercially available and open source column storage DWMS, e.g., Sybase IQ, EMC Greenplum, C-Store/Vertica, MonetDB, Sadas, FastBit, Model 204. The MOLAP implementation is based on other storage structures, like multidimensional arrays and multidimensional indexes. In this technology research is focused among others on developing efficient storage implementations, index structures, and compression techniques, e.g., [16, 17].

Query Optimizer

The way queries are executed strongly impacts the performance of a DW. The process of optimizing query executions and building robust query optimizers has been receiving substantial focus from the research community. Huge research literature exists on this issue, cf., [18]. Recently, research in this area concentrates among others on join algorithms, e.g., [19], parallel query optimization and execution, e.g., [20, 21], designing robust and more intelligent query optimizers [22].

Querying

Typically, OLAP applications analyze all data in order to deliver reliable results. Nonetheless, if volumes of data are extremely large, some researchers propose to apply various techniques to computing approximate results, like for example (1) sampling, based on histograms or wavelets, e.g., [23, 24], (2) statistical properties of data, or (3) apply a probability density function, e.g., [25].

Data Structures

Query execution plans profit from additional data structures that make data searching faster and reduce the volume of data that has to be retrieved. Different data structures have been developed and applied in practice in commercial DWMSs. Some of them include: (1) various types of indexes, like join indexes, e.g., [26], bitmap indexes, e.g., [27, 28], and bitmap join indexes, e.g., [29, 30], (2) materialized views and query rewriting techniques, e.g., [31], (3) table partitioning, e.g., [32, 33, 34]. In these areas multiple research works focus on compressing bitmap indexes, e.g., [35, 36, 37, 28], algorithms for materialized view selection and fast refreshing, e.g., [38, 31, 39, 40], and finding the most efficient partitioning schemes, e.g., [41, 42].

Testing Performance

An important practical issue concerns the ability of evaluating the performance of a DWMS and compare multiple architectures with this respect.

To this end, multiple approaches to testing the performance and robustness of a DWMS have been developed, e.g., [43, 44, 45, 46] and are being further intensively investigated.

2.1.2 Chapter Focus

In this chapter we overview a narrow area only of the huge research and technological area devoted to increasing a data warehouse performance. We focus on the aforementioned data structures (indexes, materialized views, and partitions) in ROLAP servers and illustrate their basic functionality and usage in commercial DWMSs, i.e., Oracle, DB2, and SQL Server.

The chapter is organized as follows. In Section 2.2 we present basic data warehouse concepts and an example data warehouse used throughout this chapter. In Section 2.3 we present index data structures applied to the optimization of star queries. In Section 2.4 we discuss a technique of materializing query results and using the materialized results for query optimization. In Section 2.5 we present table partitioning techniques. Finally, Section 2.6 includes the chapter summary.

2.2 Basic Concepts

2.2.1 DW Model and Schema

In order to support various analyses, data stored in a DW are represented in a *multidimensional data model* [47, 48]. In this model an elementary information being the subject of analysis is called a *fact*. It contains numerical features, called *measures* that quantify the fact. Values of measures are analyzed in the context of *dimensions*. Dimensions often have a hierarchical structure composed of levels, such that $L_i \rightarrow L_j$, where $\rightarrow$ denotes hierarchical assignment between a lower level L_i and upper level L_j, also known as a roll-up or an aggregation path [49]. Following the aggregation path, data can be aggregated along a dimension hierarchy.

The multidimensional model is often implemented in relational OLAP servers (ROLAP) [50], where fact data are stored in a *fact table*, and level data are stored in *dimension tables*. In a ROLAP implementation two basic types of conceptual schemas are used, i.e. a star schema and a snowflake schema [50]. In the *star schema* each dimension is composed of only one (typically denormalized) level table. In the *snowflake schema* each dimension is composed of multiple normalized level tables connected by foreign key - primary key relationships.

An example star schema that will be used throughout this chapter is shown in Figure 2.1. It is composed of the *Sales* fact table, and three dimension tables, namely *Products*, *Customers*, and *Time*. The *Sales* fact table is connected with its dimension table via three foreign keys, namely *ProductID*, *CustomerID*, and *TimeKey*. The fact table includes measure column *SalesPrice*.

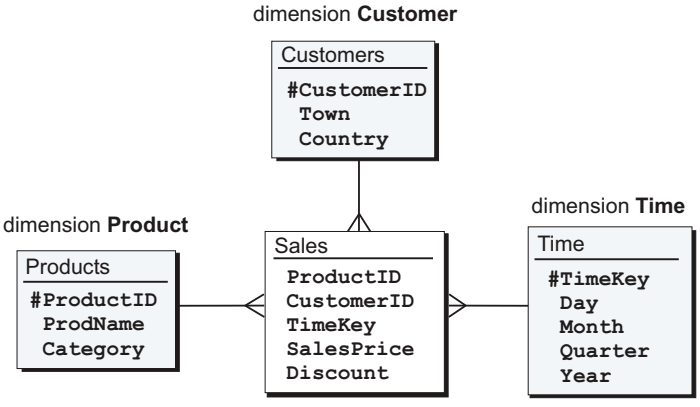


Fig. 2.1. Example data warehouse star schema on sales of products

2.2.2 Star Queries

Based on a DW schema, analytical queries are executed in a DW. As mentioned earlier, such queries typically join multiple tables, filter and sort data, as well as aggregate data at different levels of dimension hierarchies. Typically, the queries join a fact table with multiple dimension tables and are called *star queries*.

An example star query computing the yearly sales (in years 2009 and 2010) of products belonging to category 'electronic' in countries of sales, is shown below.

```
select sum(SalesPrice), ProdName, Country, Year
from Sales s, Products p, Customers c, Time t
where s.ProductID=p.ProductID
and s.CustomerID=c.CustomerID
and s.TimeKey=t.TimeKey
and p.Category in ('electronic')
and t.Year in (2009, 2010)
group by ProdName, Country, Year;
```

2.3 Index Data Structures

Star queries can profit from applying some indexes in the process of retrieving data. Three indexes, the most frequently used in practice include: a join index, a bitmap index, and a bitmap join index, which are outlined in this section.

2.3.1 Join Index

A *join index* represent the materialized join of two tables, say R and S. As defined in [51, 26], a join index is a table composed of two attributes. It stores

the set of pairs (r_i, s_j) where r_i and s_j denote tuple identifiers from R and S, respectively, that join on a given predicate. In order to make searching the join index faster, it is assumed that the join index is physically ordered (clustered) by one of the attributes. Alternatively, the access to the join index can be organized by means of a B-tree or a hash index [30].

A join index can be created either on a join attribute or on a non-join attribute of a table. In the second case, the index is organized in such a way that it allows lookups by the value of the non-joined attribute in order to retrieve the ROWIDs of rows from the joined tables that join with the non-joined attribute value.

In the context of a data warehouse, a join index is applied to joining a dimension table and a fact table. The index is created either on a join attribute of a dimension table or on another attribute (typically storing unique values) of a dimension table. In order to illustrate the idea behind the join index let us consider the example below.

Example 1. Let us consider tables *Products* and *Sales* from the DW schema shown in Figure 2.1. Their content is shown in Table 2.1. For clarity purpose, both tables include also explicit column *ROWID* that stores physical addresses of records and that serve as row identifiers.

Table 2.1. Example tables in the star schema on sales of products (from Fig.2.1)

Sales			
ROWID	SalesPrice	Discount	ProductID
0AA0	...	5	100
0AA1	...	15	230
0AA2	...	5	100
0AA3	...	10	300
0AA4	...	10	300
0AA5	...	15	230

Products			
ROWID	ProductID	ProdName	Category
BFF1	100	HP Pavilion	electronic
BFF2	230	Dell Inspiron	electronic
BFF3	300	Acer Ferrari	electronic

The join index defined on column *ProductID* is shown in Table 2.2.

Table 2.2. Example join index on ProductID

Products.ROWID	Sales.ROWID
BFF1	0AA0
BFF1	0AA2
BFF2	0AA1
BFF2	0AA5
BFF3	0AA3
BFF3	0AA4

As one can observe from the above example, the join index stores a materialized (precomputed) join of the *Products* and *Sales* tables. Thus, it will optimize queries like:

```
select ...
from Sales s, Products p
where s.ProductID=p.ProductID ...
```

2.3.2 Bitmap Index

OLAP queries not only join data, but also filter data by means of query predicates. Efficient filtering of large data volumes is well supported by the so-called bitmap indexes [52, 27, 28, 53]. Conceptually, a *bitmap index* created on attribute a_m of table T is organized as the collection of bitmaps. For each value val_i in the domain of a_m a separate bitmap is created. A bitmap is a vector of bits, where the number of bits equals to the number of records in table T. The values of bits in bitmap for val_i are set as follows. The n-th bit is set to 1 if the value of attribute a_m for the n-th record equals to val_i. Otherwise the bit is set to 0.

In order to illustrate the idea behind the bitmap index let us consider the example below.

Example 2. Let us review the *Sales* fact table, shown in Table 2.3. The table contains attribute *Discount* whose domain includes three values, namely *5*, *10*, and *15*, denoting a percent value of discount. A bitmap index created on this attribute will be composed of three bitmaps, noted as *Bm5perc*, *Bm10perc*, and *Bm15perc*, respectively, as shown in Table 2.3.

The first bit in bitmap *Bm15perc* equals to 0 since the *Discount* value of the first record in table *Sales* does not equal *15*. The second bit in bitmap *Bm15perc* equals to 1 since the *Discount* value of the second record in table *Sales* equals to *15*, etc.

Such bitmap index will offer a good response time for a query selecting for example data on sales with 5% or 15% discounts. In order to find sales records fulfilling this criterion, it is sufficient to OR bitmaps *Bm5perc* and *Bm15perc* in order to construct a result bitmap. Then, records pointed to by bits equal to '1' in the result bitmap are fetched from the *Sales* table. □

At the implementation level, bitmap indexes are organized either as B-trees [27] or as simple arrays in a binary file [54]. In the first case, B-tree leaves store bitmaps and a B-tree is a way to organize indexed values and bitmaps in files.

Bitmap indexes allow to answer queries with the `count` function without accessing tables, since answers to such queries can be computed by simply counting bits equaled to '1' a result bitmap. Moreover, such indexes offer a good query performance for attributes of narrow domains. For such attributes, a bitmap index will be much smaller than a traditional B-tree

Table 2.3. Example bitmap index created on the *Discount* attribute

Sales			bitmap index on *Discount*		
SalesPrice	Discount	...	Bm5perc	Bm10perc	Bm15perc
...	5	...	1	0	0
...	15	...	0	0	1
...	5	...	1	0	0
...	10	...	0	1	0
...	10	...	0	1	0
...	15	...	0	0	1

index. Additionally, while evaluating queries with multiple predicates with equality and inequality operators, a system processes bitmaps very fast by AND-ing/OR-ing them.

The size of a bitmap index strongly depends on the cardinality (domain width) of an indexed attribute, i.e., the size of a bitmap index increases when the cardinality of an indexed attribute increases. Thus, for attributes of high cardinalities (wide domains) bitmap indexes become very large (much larger than a B-tree index). As a consequence, they cannot fit into main memory and the efficiency of accessing data with the support of such indexes deteriorates [55].

In order to reduce the size of bitmap indexes defined on attributes of high cardinalities, two following approaches have been proposed in the research literature, namely: (1) extensions to the structure of the basic bitmap index, and (2) bitmap index compression techniques. In the first approach, two main techniques, generally called binning as well as bit slicing, can be distinguished.

Extensions to the Structure of the Basic Bitmap Index

In [56] (called *range-based bitmap indexing*) and in [57, 58, 59] (called *binning*), values of an indexed attribute are partitioned into ranges. A bitmap is constructed for representing a given range of values, rather than a distinct value. Bits in a single bitmap indicate whether the value of a given attribute of a row is within a specific range. This technique can also be applied when values of an indexed attribute are partitioned into sets.

The technique proposed in [60] can be classified as a more general form of binning. In [60] sets of attribute values are represented together in a bitmap index. Such a technique reduces storage space for attributes of high cardinalities. The selection of attribute values represented in this kind of an index is based on query patterns and their frequencies, as well as on the distribution of attribute values.

Another form of binning was proposed in [52]. This technique, called *property maps*, focuses on managing the total number of bins assigned to all indexed attributes. A property map defines properties on each attribute, such

as the set of queries using the attribute, distribution of values for the attribute, or encoded values of the attribute. The properties are represented as vectors of bits. A query processor needs extension in order to use property maps. Property maps support multi-attribute queries, inequality queries or high selectivity queries, and they are much smaller than bitmap indexes.

The second technique is based on the so-called *bit-sliced index* [61, 62, 55]. It is defined as an ordered list of bitmaps, $B^n, B^{n-1}, \ldots, B^1, B^0$, that are used for representing values of a given attribute A, i.e., B^0 represents the 2^0 bit, B^1 represents the 2^1 bit, etc. Every value of an indexed attribute is represented on the same number of n bits. As a result, the encoded values in a table form n bitmaps. The bitmaps are called bit-slices. Data retrieval and computation are supported either by the bit-sliced index arithmetic [63] or by means of a dedicated retrieval function [55]. Additionally, a mapping data structure is required for mapping the encoded values into their real values [55].

Compression Techniques

The second approach that allows to reduce the size of a bitmap index defined on an attribute of high cardinality is based on compression. Four main loss-less techniques can be distinguished in the research literature, namely: (1) *Byte-aligned Bitmap Compression* (BBC) [64], (2) *Word-Aligned Hybrid* (WAH) [28, 65, 53, 66], (3) Position List WAH (PLWAH) [67, 35], and (4) RL-Huffman [36] and RLH [68, 37].

All the aforementioned compression techniques apply the *run-length* encoding. The basic idea of the run-length encoding consists in encoding continuous vectors of bits having the same value (either "0" or "1") into: (1) a common value of all bits in the vector (i.e., either "0" for a vector composed of zeros or "1" for a vector composed of ones) and (2) the length of the vector (i.e., the number of bits having the same value). Before encoding, a bitmap is divided into words. Next, words are grouped into the so-called runs. The *run* is composed of words that can be either a fill or a tail. The *fill* represents series of words that are composed of bits of the same value. The *tail* represents the series of words that are composed of both "0" and "1" bits. Fills are compressed because of their homogeneous content, whereas tails are not.

BBC divides bit vectors into 8-bit words, WAH and PLWAH divide them into 31-bit words, RLH uses words of a parameterized length, whereas RL-Huffman does not divide a bitmap into words. PLWAH is the modification of WAH. PLWAH improves compression if tail T that follows fill F differs from F on few bits only. In such a case, the fill word encodes the difference between T and F on some dedicated bits. Moreover, BBC uses four different types of runs, depending on the length of a fill and the structure of a tail, whereas the other compression techniques use only one type of a run. The overall idea of the WAH compression is illustrated with the example below taken from [28].

Example 3. For the sake of simplicity let us assume that a 32-bit processor is used. A bitmap being compressed is composed of 5456 bits, as shown in Figure 2.2a. The WAH compression of the bitmap is executed in three following steps.

In the first step, the bitmap is divided into groups composed of 31 bits, as shown in Figure 2.2b. In the example, 176 such groups are created. In the second step, adjacent groups containing identical bits are merged into one group, as shown in Figure 2.2c. Since *group 1* is heterogeneous, i.e., it is composed of "0" and "1" bits, it is not merged with a group following it. Groups 2 to 175 are homogeneous (composed of "0" bits) and they are merged into one large group, denoted in Figure 2.2c as *group 2-175*. This group includes 174*31 bits. The last *group 176*, similarly as *group 1*, is heterogeneous and it cannot be merged with a group preceding it. As the result of group merging, three final groups are created, as shown in Figure 2.2c.

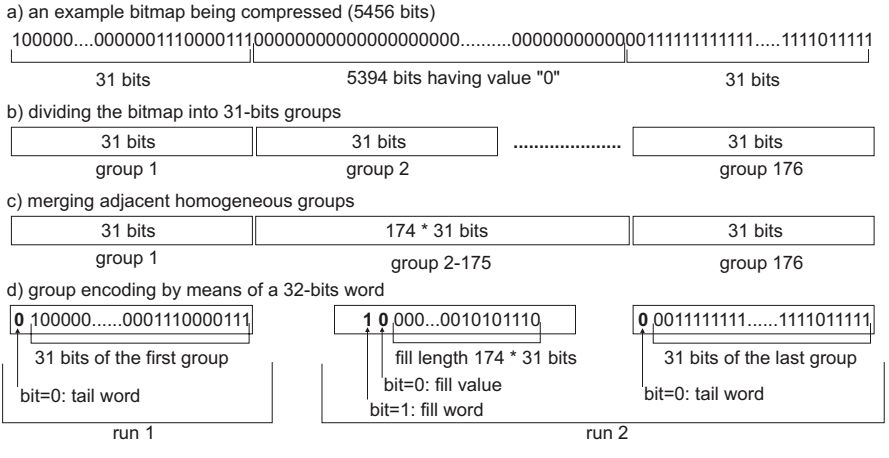


Fig. 2.2. The steps of the WAH compression

In the third step, the three final groups are encoded on 32-bit words as follows (cf. Figure 2.2d). The first group represents the tail of the first run. The most significant bit (the leftmost one) has value "0" denoting a tail. Next 31 bits are original bits of *group 1*. The second group (*group 2-175*) represents the fill of the second run. The most significant bit (at position ^{31}) is set to "1" denoting a fill. The bit at position 2^{30} is set to "0" denoting that all bits in original *group 2-175* have value "0", i.e., the fill is used for compressing groups whose all bits have value "0". The remaining 30 bits are used for encoding the number of homogeneous groups filled with "0." In the example, there are 174 such groups. The number of homogeneous groups is represented by the binary value equaled to 00000000000000000000010101110, stored on the remaining 30 bits. The last 31-bits group, denoted as *group 176*, represents

the tail of the second run. The most significant bit in this group has value "0" denoting a tail. The remaining 31 bits are original bits of *group 176*. □

The compression techniques proposed in [36] and [37] additionally apply the Huffman compression [69] to the run-length encoded bitmaps. The main differences between [36] and [37] are as follows. First, in [36] only some bits in a bit vector are of interest, the others, called 'don't cares' can be replaced either by zeros or ones, depending on the values of neighbor bits. In RLH all bits are of interest and have their exact values. Second, in [36] the lengths of homogeneous subvectors of bits are counted and become the symbols that are encoded by the Huffman compression. RLH uses run-length encoding for representing distances between bits having value 1. Next, the distances are encoded by the Huffman compression.

In practice, the commercial systems (Oracle DBMS) and prototype systems (FastBit) [70, 71, 54] apply bitmap compression techniques. Oracle uses the BBC compression whereas FastBit uses the WAH compression.

Bitmap Indexes in Oracle

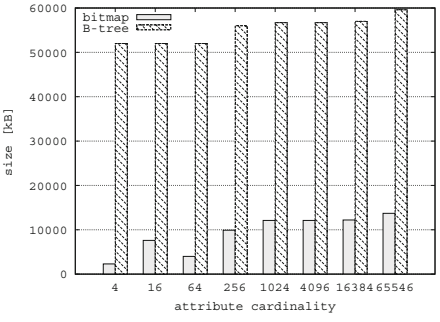
The Oracle DBMS [72] supports explicitly created bitmap indexes. A bitmap index is created by means of the below command.

```
create bitmap index BIName on table(column);
```

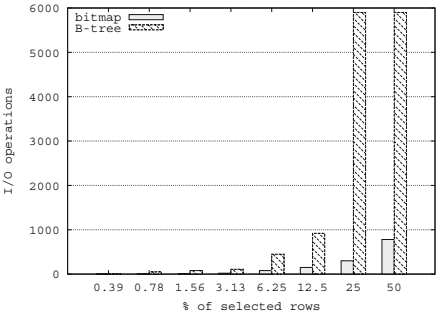
As mentioned in Section 2.3.2, the size of a bitmap index increases with the increase of the cardinality of the indexed attribute. In order to reduce the size of such a bitmap index, Oracle DBMS applies a compression based on BBC. When the sparsity of bitmaps exceeds a given threshold, Oracle automatically compresses bitmaps. Figure 3(a) shows how the size of a bitmap index depends on the cardinality of the indexed attribute. The cardinality was parameterized from 4 to 65564 unique values. The experiment was conducted on Oracle10g R2. The indexed table included 320 000 000 of rows. As a reference, the size of the Oracle B*-tree is also included in the chart.

As we can observe from chart 3(a), with the increase of cardinality from 4 to 16 the size of the bitmap index increases. Next, for cardinality equal to 64, the size of the BI decreases that is caused by compressing the BI. Next, with the increase of the cardinality above 64 we observe further increase in the size of the compressed BI.

Bitmap indexes support fast computations of function `count`. Figure 3(b) shows the computation efficiency of `count(1)` for various selectivities of a query. The query selected from 0.39% to 50% of rows based on the values of the indexed attribute. As a reference, the figure includes also the performance of the Oracle B*-tree index. The characteristics of the indexes were tested on identical attributes. It must be noticed that when `count` is replaced by other aggregate functions, like for example `sum`, `avg`, the bitmap index performs much worse, but still better than B*-tree. It is because, in order to compute these aggregate functions data must be fetched from disk.



(a) The size of a bitmap index and a B*-tree index



(b) The computation efficiency of `count(1)` with the support of a bitmap index and a B*-tree index

Fig. 2.3. Comparison of some features of a bitmap index and a B*-tree index in Oracle (# of rows in an indexed table: 320 000 000)

Bitmap Indexes in DB2

IN the DB2 DBMS, queries with predicates on multiple indexed attributes can be optimized by means of the *index AND-ing* technique. In general, this technique is based on determining the intersection of the sets of ROWIDs, which are retrieved with the support of individual indexes on attributes used in query predicates. The intersection is determined with the support of bloom filters [73, 74]. The bloom filters transform each set of ROWIDs into a separate temporal bitmap. Multiple bitmaps can then be further processed by AND-ing or OR-ing them, depending on query predicates. The bitmaps are called *dynamic bitmap indexes*.

As an example illustrating the application of a dynamic bitmap index to the optimization of a query on one table, let us consider table *Sales* with two B-tree indexes, one defined on attribute *ProductID* and one on *CustomerID*. An example query with predicate `ProductID in ('ThE1', 'SoV1', 'DeV1') and CustomerID = 100` could be answered by applying the index AND-ing technique with the support of a dynamic bitmap index as follows. First, by using index on *ProductID* a bitmap is created that describes rows fulfilling the predicate `ProductID in ('ThE1', 'SoV1', 'DeV1')`. Let us denote the bitmap as B_{Prod}. Similarly, by using index on *CustomerID* a bitmap, say B_{Cust} is created that describes rows fulfilling the predicate `CustomerID = 100`. Second, the final bitmap is computed by AND-ing B_{Prod} and B_{Cust}. Based on the final bitmap that is transformed back to ROWIDs, rows fulfilling the predicates are retrieved.

As an example illustrating the application of dynamic bitmap indexes to optimizing star queries, let us consider star query $Q1$ from Section 2.3.3. In order to use the technique, a B-tree index has to be created on each of the foreign keys. In the first step, every dimension table is semi-joined with the *Sales* table in order to determine the set of *Sales* rows that join with rows form each of the dimensions. Let the result of $Sales \ltimes Products$, i.e., the set of ROWIDs fulfilling the predicate on *ProdName* is denoted as S_{Prod}. Let the result $Sales \ltimes Customers$, i.e., the set of ROWIDs fulfilling the predicate on *Town* is denoted as S_{Cust}. Finally, let $Sales \ltimes Time$, i.e., the set of ROWIDs fulfilling the predicate on *Year* is denoted as S_{Time}. In the second step, S_{Prod}, S_{Cust}, and S_{Time} are transformed to three bitmaps, denoted as B_{Prod}, B_{Cust}, and B_{Time}, respectively. In the third step, bitmaps B_{Prod}, B_{Cust}, B_{Time} are AND-ed. The final bitmap describes rows fulfilling the whole query predicate. It is then transformed back to ROWIDs. Based on the ROWIDs records are fetched from the *Sales* table. If some columns from dimension tables are needed in the query result, then the final set of sales rows is joined with appropriate rows from the dimensions.

Bitmap Indexes in SQL Server

Similarly as DB2, SQL Server [75] does not support explicitly created bitmap indexes. Instead, it supports bitmap filters (this mechanism is available from version 2005). However, bitmap filters are not bitmap indexes. The *bitmap filter* represents (in a compact format) the set of values from one table being joined, typically a dimension table. Based on the bitmap filter, rows from the second joined table, typically a fact table, are filtered. Thus, the bitmap filter is applied as a semi-join reduction technique but only in parallel execution plans. A bitmap filter is automatically created by the SQL Server query optimizer when the optimizer estimates that such a filter is selective. A bitmap filter is a main memory data structure.

In order to illustrate the application of bitmap filters to the optimization of a star query, let us consider star query $Q1$ from Section 2.3.3. It could be

processed as follows. In the first step, bitmap filters are created. Bitmap filter BF_{Prod} is created, based on values ('ThinkPad Edge', 'Sony Vaio', 'Dell Vostro') from the *Products* dimension table. In parallel, bitmap filters on dimension tables *Customers* and *Time* are created, based on the predicates `Town='London'` and `Year=2009`, respectively. Let us denote the filters as BF_{Cust} and BF_{Year}, respectively. In the second step, BF_{Prod} is used for semi-join reduction with the *Sales* table (let denote the intermediate result as temporary table $Sales^{reduced}_{Prod}$). In the third step, bitmap filter BF_{Cust} is applied as semi-join reduction operator to temporary table $Sales^{reduced}_{Prod}$. Let us denote its result as $Sales^{reduced}_{Prod\_Cust}$. In the fourth step bitmap filter BF_{Year} is applied to $Sales^{reduced}_{Prod\_Cust}$, resulting in temporary table $Sales^{reduced}_{Prod\_Cust\_Year}$. Finally, rows fulfilling the whole predicate of the query are fetched by means of $Sales^{reduced}_{Prod\_Cust\_Year}$. The order in which the bitmap filters are applied depends on the selectivity of the filters. The most selective one should be applied first.

2.3.3 Bitmap Join Index

The advantages of the join index and the bitmap index have been combined in a *bitmap join index* [76, 30, 62]. This index, conceptually is organized as the join index but the entries to the bitmap join index are organized as a lookup by the value of a dimension attribute. Each entry to the bitmap join index is composed of the ROWID of a row from a dimension table (or an attribute uniquely identifying a row in a dimension table) and a bitmap (possibly compressed) describing rows from a fact table that join with this value. Similarly as for the join index, the access to the bitmap join index lookup column can be organized by means of a B-tree or a hash index.

In order to illustrate the idea behind the bitmap join index let us consider the example below.

Example 4. Let us return to Example 1 and let us define the bitmap join index on attribute *ProductID* of table *Products*. Conceptually, the entries of this index are shown in Table 2.4. The lookup of the index is organized by the values of attribute *ProductID*. Assuming that the leftmost bit in each of the bitmaps represents the first row in the *Sales* table, one can see that the first and third sales row join with product identified by value 100, for example.

Table 2.4. Example bitmap join index organized as a lookup by attribute *Products.ProductID*

Products.ProductID	bitmap
100	1 0 1 0 0 0
230	0 1 0 0 0 1
300	0 0 0 1 1 0

Every indexed value (e.g. 100) in dimension table *Products* has associated a bitmap describing records from fact table *Sales* that join with the dimension record. □

The bitmap join index takes the advantage of the join index since it allows to materialize a join of tables. It also takes the advantage of the bitmap index with respect to efficient AND, OR, NOT operations on bitmaps.

Bitmap Join Indexes in Oracle

Oracle also supports the *bitmap join index* that offers a very good performance of star queries. As an example let us consider the below star query $Q1$, that joins the *Sales* fact table with all of its dimension tables.

```
/* query Q1 */
select sum(SalesPrice)
from Sales, Products, Customers, Time
where Sales.ProductID=Products.ProductID
and Sales.CustomerID=Customers.CustomerID
and Sales.TimeKey=Time.TimeKey
and ProdName in ('ThinkPad Edge', 'Sony Vaio', 'Dell Vostro')
and Town='London'
and Year=2009;
```

In order to reduce the execution time of $Q1$, three bitmap indexes can be created, as shown below.

```
create bitmap index BI_Pr_Sales
on Sales(Products.ProdName)
from Sales, Products
where Sales.ProductID=Products.ProductID;

create bitmap index BI_Cu_Sales
on Sales(Customers.Town)
from Sales, Customers
where Sales.CustomerID=Customers.CustomerID;

create bitmap index BI_Ti_Sales
on Sales(Time.Year)
from Sales, Time
where Sales.TimeKey=Time.TimeKey;
```

The query execution plan, shown below, reveals how the bitmap join indexes are used. First, bitmaps for 'ThinkPad Edge', 'Sony Vaio', and 'Dell Vostro' are retrieved and OR-ed (cf. lines 10-12), creating a temporary bitmap. Next, the bitmap for 'London' is retrieved (cf. line 8) and it is AND-ed with the temporary bitmap (cf. line 7). The result bitmap is then converted to ROWIDs of records fulfilling the criteria (cf. line 6). In this query plan table *Time* was accessed with the support of a B-tree index defined on its primary key, rather than by means of the bitmap join index.

```
-------------------------------------------------------------------------------------
|Id  | Operation                      |Name       |Rows |Bytes |Cost(%CPU)|Time    |
-------------------------------------------------------------------------------------
| 0 | SELECT STATEMENT                |           |    1|   58 |   13  (8)|00:00:01|
| 1 |  SORT AGGREGATE                 |           |    1|   58 |          |        |
| 2 |   NESTED LOOPS                  |           |   21| 1218 |   13  (8)|00:00:01|
| 3 |    HASH JOIN                    |           |   22| 1012 |   12  (9)|00:00:01|
| 4 |     TABLE ACCESS FULL           |PRODUCTS   |    3|   51 |    3  (0)|00:00:01|
| 5 |     TABLE ACCESS BY INDEX ROWID |SALES      | 1155|33495 |    8  (0)|00:00:01|
| 6 |      BITMAP CONVERSION TO ROWIDS|           |     |      |          |        |
| 7 |       BITMAP AND                |           |     |      |          |        |
| 8 |        BITMAP INDEX SINGLE VALUE|BI_CU_SALES|     |      |          |        |
| 9 |        BITMAP OR                |           |     |      |          |        |
|10 |         BITMAP INDEX SINGLE VALUE|BI_PR_SALES|     |      |          |        |
|11 |         BITMAP INDEX SINGLE VALUE|BI_PR_SALES|     |      |          |        |
|12 |         BITMAP INDEX SINGLE VALUE|BI_PR_SALES|     |      |          |        |
|13 |    TABLE ACCESS BY INDEX ROWID  |TIME       |    1|   12 |    1  (0)|00:00:01|
|14 |     INDEX UNIQUE SCAN           |PK_TIME    |    1|      |    0  (0)|00:00:01|
-------------------------------------------------------------------------------------
```

Alternatively, it is possible to create one bitmap join index on all of the dimension attributes. The example below command creates the bitmap join index on three attributes of the dimension tables, namely: *Products.prodName*, *Customers.Town*, and *Time.Year*.

```
create bitmap index BI_Pr_Cu_Ti_Sales
on Sales(Products.ProdName, Customers.Town, Time.Year)
from Sales, Products, Customers, Time
where Sales.ProductID=Products.ProductID
and Sales.CustomerID=Customers.CustomerID
and Sales.TimeKey=Time.TimeKey;
```

With the support of this index, the aforementioned star query $Q1$ is executed as shown below. As we can observe, its execution plan is much simpler than before. The bitmap join index is accessed only once (cf. line 5).

```
-----------------------------------------------------------------------------------------
|Id | Operation                      |Name              |Rows |Bytes|Cost(%CPU)|Time     |
-----------------------------------------------------------------------------------------
| 0 | SELECT STATEMENT               |                  |    1|   29|    7  (0)|00:00:01 |
| 1 |  SORT AGGREGATE                |                  |    1|   29|          |         |
| 2 |   INLIST ITERATOR              |                  |     |     |          |         |
| 3 |    TABLE ACCESS BY INDEX ROWID |SALES             |   22|  638|    7  (0)|00:00:01 |
| 4 |     BITMAP CONVERSION TO ROWIDS|                  |     |     |          |         |
| 5 |      BITMAP INDEX SINGLE VALUE |BI_PR_CU_TI_SALES|     |     |          |         |
-----------------------------------------------------------------------------------------
```

Unfortunately, this bitmap join index cannot be used for answering queries with predicates on one or two dimensions, like for example

```
ProdName in ('ThinkPad Edge', 'Sony Vaio', 'Dell Vostro') and Town=
'London'
```

or

```
Town='London' and Year=2009
```

or

```
Year=2009
```

On the contrary, the three independent bitmap join indexes defined earlier, i.e., *BI_Pr_Sales*, *BI_Cu_Sales*, *BI_Ti_Sales* offer much flexible indexing scheme as they can be used answering queries with predicates on some of the dimensions.

2.4 Materialized Views and Query Rewriting

Execution time of complex, time consuming star queries can be reduced by physically storing and re-using their previously computed results. This way, a precomputed result is perceived by a query optimizer as another source of data that can be queried. The precomputed result is commonly called a *materialized view* (MV). If a user executes query Q that computes values that have already been stored in materialized view MV_i, then a query optimizer will rewrite original query Q into another query Q' so that Q' is executed on MV_i and Q' returns the same result as original query Q. This technique of automatic, in-background rewriting of users' queries is called a *query rewriting*. Notice that a user's query must not necessarily be exactly the same as the query whose result was stored in a materialized view. In this case, a query optimizer may join a materialized view with other MVs or tables, may further aggregate and filter the content of a MV and may project columns of a MV, in order to answer a user's query [77]. In order to illustrate the idea behind the query rewriting mechanism let us consider the example below.

Example 5. Let us return to the DW logical schema shown in Figure 2.1 and let us assume that in this DW users execute the below queries. The first one computes quarterly sales of product categories in countries. The second one computes monthly and quarterly sales of products belonging to category 'electronic'.

```
select Category, Country, Quarter,
       sum(SalesPrice) as SumSales
from Sales s, Products p, Customers c, Time t
where s.ProductID=p.ProductID
and s.CustomerID=c.CustomerID
and s.TimeKey=t.TimeKey
group by Category, Country, Quarter;

select ProdName, Country, Month,
       sum(SalesPrice) as SumSales
from Sales s, Products p, Customers c, Time t
where s.ProductID=p.ProductID
and s.CustomerID=c.CustomerID
and s.TimeKey=t.TimeKey
and p.Category='electronic'
group by ProdName, Country, Month
```

Both queries could be optimized by materialized view *SalesMV1*, shown below, whose query makes available monthly and quarterly sales values of

products and their categories for each country. Both of the above queries could be answered by means of the materialized view by further aggregating its content.

```
create materialized view SalesMV1
...
select ProdName, Category, Country, Month, Quarter,
       sum(SalesPrice) as SumSales
from Sales s, Products p, Customers c, Time t
where s.ProductID=p.ProductID
and s.CustomerID=c.CustomerID
and s.TimeKey=t.TimeKey
group by ProdName, Category, Country, Month, Quarter
```
□

As we can see from the example, the definition of a materialized view includes a query. Tables referenced in the query are called *base tables*.

Although in the above example one materialized view is used for rewriting two queries, in practice, multiple materialized views need to be created in order to optimize a certain workload of queries. For this reason, a challenging research issue concerns the selection of such a set of materialized views that: (1) will be used for optimizing the greatest possible number of the most expensive queries and (2) whose maintenance will not be costly. Several research works have addressed this problem and they have proposed multiple algorithms for selecting optimal sets of materialized views for a given query workload, e.g. [78, 79, 80, 81]. Commercial DB/DWMSs, like, Oracle, DB2, and SQL Server provide tools that analyze workloads and based on them, propose sets of database objects, typically materialized views and indexes, for optimizing the workloads.

2.4.1 Materialized Views in Oracle

In Oracle, a materialized view is created with the `create materialized view` command. Its definition includes: (1) the moment when the view is filled in with data, (2) its refreshing method (fast, complete, force), (3) whether the materialized view is refreshed automatically or on demand, (4) row identification method (required for incremental refreshing), (5) the view automatic refreshing interval, and (6) a query computing the content of the materialized view.

A command creating an example materialized view *YearlySalesMV* is shown below.

```
create materialized view YearlySalesMV1
build immediate
refresh force
with rowid
as
select ProdName, Category, Quarter, Year,
```

```
       sum(SalesPrice) as SumSales
from Sales s, Products p, Time t
where s.ProductID=p.ProductID
and s.TimeKey=t.TimeKey
group by ProdName, Category, Quarter, Year;
```

The `build immediate` clause denotes that the view will be filled in with data during the execution of the command. The `refresh force` clause denotes that the system automatically selects the refreshing mode (either fast, i.e., incremental or complete). If the system is able to refresh the MV fast, then this method is used. Otherwise the complete refreshing is used. The `with ROWID` clause defines the method of identifying rows in the MV and its base tables. In this example, rows will be identified based on their physical addresses (ROWIDs). Rows can also be identified based on their primary keys. To this end, the `with primary key` clause is applied. Row identification is required for incremental refreshing.

For some versions of Oracle (e.g., 10g) clause `with rowid` clashes with clause `enable query rewrite`. The latter makes available a MV for query rewriting. Normally, this clause is part of the MV definition, placed between `with rowid|primary key` and `as`. In cases these clauses clash, one has to make the view available for query rewriting by executing additional command shown below.

```
alter materialized view MVName enable query rewrite;
```

Oracle materialized views will be used in query rewriting provided that: (1) the cost-based query optimizer is used, (2) materialized views have been enabled for query rewriting, (3) the system works in the query rewriting mode, and (4) a user executing a query has appropriate privileges.

In order to illustrate the query rewriting process, let us consider the below query. We assume that MV *YearlySalesMV1* was created and made available for query rewriting. The query execution plan reveals that it was automatically rewritten on the *YearlySalesMV*, cf. row number 2 in the below plan. This row denotes that the MV was sequentially scanned and its content grouped (cf. row number 1) in order to compute the more coarse aggregate.

```
select ProdName, Year, sum(SalesPrice) as SumSales
from Sales s, Products p, Time t
where s.ProductID=p.ProductID
and s.TimeKey=t.TimeKey
and t.Year=2009
group by ProdName, Year;
```

```
-------------------------------------------------------------------------------------
|Id |Operation                    |Name          |Rows |Bytes|Cost(%CPU)|Time    |
-------------------------------------------------------------------------------------
| 0 |SELECT STATEMENT             |              | 159 | 3180|   4  (25)|00:00:01|
| 1 | HASH GROUP BY               |              | 159 | 3180|   4  (25)|00:00:01|
| 2 |  MAT_VIEW REWRITE ACCESS FULL|YEARLYSALESMV | 159 | 3180|   3   (0)|00:00:01|
-------------------------------------------------------------------------------------
```

As mentioned before, a MV can be refreshed fast, provided two conditions are fulfilled. First, the MV query definition includes only the allowed constructs. The set of these constructs is extended from version to version of the Oracle DBMS (cf. [72]). Second, each of the MV base tables has associated its own *materialized view log*. The log records DML operations applied to the base table. The content of the log is used for fast refreshing the MV.

The materialized view log is created by the below command (only its basic form is shown). For a MV created with the `with primary key` clause the MV log must include the same clause. The same applies to the `with rowid` clause. In the first case the log will store the values of columns that constitute the primary key of the table, whereas in the second case it will store the values of ROWID. A MV log may include both clauses if it serves for both types of MV. Apart from the primary key and ROWID column, a materialized view log can store values of other columns whose changing values have impact on the content of the MV. The list of these columns is denoted as `col1, ..., coln`.

```
create materialized view log on BaseTable
with primary key|ROWID (col1, ..., coln),
      sequence
      including new values;
```

Including the `sequence` clause in a materialized view log definition results in the creation of a column *SEQUENCE$$* in the log. Its values represent the order in which DML operations are executed on the base table for which the log was created. Oracle advises to create this column for fast refreshing after mixed insert, update, delete operations on the base table. The `including new values` clause forces Oracle to store in the MV log old values of columns listed in `col1, ..., coln` as well as their new values. Omitting this clause results in storing only the old values of columns. This clause must be included in the MV log definition in order to support fast refreshing MVs computing aggregates.

2.4.2 Materialized Views in DB2

In DB2 a materialized view, called a *materialized query table* (MQT) or a *summary table*, is created with the `create table` command with additional clauses defining its maintenance. The clauses include: (1) `maintained by {system | user}`, (2) `refresh {immediate | deferred}`, (3) `data initially {immediate | deferred}`, and (4) `enable query optimization`.

A MQT can be maintained either by a system or by a user. In the first case, the `maintained by system` clause is used. This is a default clause. For a *system-maintained* MQT, one can define either automatic or non-automatic refreshing mode. In the automatic mode, a MQT is refreshed automatically as the result of changes in the content of its base tables. To this end, the

definition of a MQT must include the `refresh immediate` clause. This refreshing mode requires that a unique key from each base table is included in the `select` command defining the MQT. `refresh deferred` denotes that a MQT has to be explicitly refreshed by explicit execution of the `refresh table` command, with the syntax shown below.

```
refresh table TableName {incremental|not incremental}
```

The `data initially immediate` clause causes that the content of a MQT is computed as part of the command creating the MQT. As a result, as soon as the command is finished, the MQT is filled in with its materialized data. The `data initially deferred` clause denotes that the content of the MQT is not computed as part of the command creating the MQT. After being created, the MQT is in the *check pending* state. In this state, the MQT cannot be queried until the `set integrity` command has been executed on the MQT.

The `enable query optimization` clause, similarly as in Oracle, makes a MQT available for query rewriting. It is the default clause.

An example system-maintained MQT, called *YearlySalesMV2*, is shown below. The MQT is filled in with data during its creation process (`data initially immediate`), is refreshed immediately after changes have been applied to its base tables (`refresh immediate`), and is made available for query optimization (`enable query optimization`).

```
create table YearlySalesMV2
as
(select ProdID, ProdName, Year, sum(salesPrice) as SumSales
 from Sales s, Products p, Time t
 where s.ProductID=p.ProductID
 and s.TimeKey=t.TimeKey
 and t.Year=2009
 group by ProdID, ProdName, Year)
data initially immediate
refresh immediate
maintained by system
enable query optimization;
```

In order to create a user-defined MQT, the `maintained by user` clause has to be included in the definition of a MQT. This type of a MQT can be refreshed only in the deferred mode and the `refresh table` command cannot be applied to such a MQT. Thus, a user is responsible for implementing the procedure of refreshing the MQT (by means of inserts, imports, triggers, etc.). An example definition of a user-maintained MQT is shown below. After being created, the MQT is in an inconsistent state and must be made consistent by executing the `set integrity` command.

```
create table YearlySalesMV3
as
(select Year, sum(salesPrice) as SumSales
```

```
 from Sales s, Time t
 where and s.TimeKey=t.TimeKey
 group by Year)
data initially deferred
refresh deferred
maintained by user

set integrity for YearlySalesMV3 materialized query
immediate unchecked
```

A MQT created with the `refreshed deferred` clause can be incrementally refreshed provided that a system maintains a log of changes that are used for refreshing the MQT. This log is called a *staging table*. A staging table is created by the `create table` command. After being created, a staging table is in an inconsistent state and must be made consistent by executing the `set integrity` command. The simple example below illustrates how to create a staging table for MQT *YearlySalesMV3*.

```
create table YearlySalesMV3_ST for YearlySalesMV3 propagate immediate
set integrity for YearlySalesMV3 staging immediate unchecked
```

2.4.3 Materialized Views in SQL Server

In SQL Server, a materialized view, called *indexed view*, is created by creating a unique clustered index on a view (cf. [75]). The index causes that the view is materialized, i.e., the whole result set of a query defining a view is persistently stored in a database. The definition of an indexed view has to fulfill the following conditions. First, an indexed view requires a column whose value is unique. A unique index is then created on this column that results in clustering the data by this column. Second, an indexed view has to be created with the `schemabinding` clause that prevents from modifying the base tables of a view as long as the view exists. Third, all the view base tables must be referenced by `schemaname.tablename`. Fourth, the query defining a materialized view may not contain the following SQL constructs, among others: `exists`, `not exists`, `count(*)`, `min`, `max`, `top`, `union`, `outer join`, non deterministic functions (e.g., `GetDate`), and subqueries.

The below example illustrates commands creating indexed view *YearlySalesMV2*. The `create view` command defines the view with a query. The second command creates a unique cluster index on columns *ProdID*, *ProdName*, and *Year*. As a consequence, the whole result set of the query is materialized.

```
create view YearlySalesMV2
with schemabinding
as
select ProdID, ProdName, Year, sum(salesPrice) as SumSales
from Sales s, Products p, Time t
where s.ProductID=p.ProductID
```

```
and s.TimeKey=t.TimeKey
and t.Year=2009
group by ProdID, ProdName, Year

create unique clustered index Indx_ProdID
on YearlySalesMV(ProdID, ProdName, Year)
```

Similarly as in Oracle and DB2, in the Developer and Enterprise editions of SQL Server, an indexed view can be used by a query optimizer for query rewriting. In other editions of SQL Server, in order to force a query optimizer to rewrite a query based on an indexed view, the query must explicitly reference the indexed view and must include hint `noexpand`. In order to force a query optimizer to use base tables, rather than an indexed view, a query must include option `expand views`. Both of them are used in the below two query templates.

```
select Column1, Column2, ...
from Table, IndexedView with (noexpand)
where ...

select Column1, Column2, ...
from Table, IndexedView
where ...
option (expand views)
```

Indexed views, similarly as traditional indexes are automatically refreshed by a system. The refreshing mode is immediate and incremental, as for traditional indexes. Notice that, non-clustered indexes can also be created on a materialized view in order to support quicker access to the view content.

2.5 Partitioning

Partitioning is a mechanism of dividing a table or index into smaller parts, called *partitions*. Due to space limitations, in this chapter we will focus only on table partitioning.

The most benefit from partitioning is achieved if every partition is stored on a separate disc. This way, multiple partitions can be accessed in parallel without disc contest.

There are two types of partitioning, namely horizontal and vertical. When a table, say T, is partitioned *horizontally*, its content is divided into disjunctive subsets of rows. Every partition includes identical schema, that is the schema of T. When table T is partitioned *vertically*, it is divided into disjunctive subsets of columns (except a primary key that exists in every partition) and includes all rows from the initial table.

Partitioning is guided by three following rules, namely completeness, disjointness, and reconstruction. *Completeness* states that when table T was partitioned into P_1, P_2, ..., P_n then every row from T or its fragment must

be stored in one of these partition. This criterion guarantees that after partitioning no data will disappear. *Disjointness* states that when table T was partitioned into P_1, P_2, ..., P_n then every row or its fragment from T must be stored in exactly one partition. An exception to this rule is vertical partitioning where ever vertical partition stores a primary key of T. This criterion guarantees that partitioning does not introduce data redundancy. *Reconstruction* states that there mus be a mechanism of reconstructing original table T from its partitions. In horizontal partitioning the reconstruction of T is done by unions of the partitions, whereas in vertical partitioning it is done by joining the partitions.

In horizontal partitioning rows from an original table are placed in partitions based on partitioning criteria. In this type of partitioning rows are divided into subsets based on the value of a selected attribute (or attributes), called a *partitioning attribute*. With this respect, there are several techniques of partitioning rows based on a partitioning attribute. The first one is *hash-based*. In this technique, rows are placed in partitions based on a hash function that for each row takes as its argument the value of a partitioning attribute and returns the number of a partition where the row is to be stored. The second one is *range-based*. In this technique, every partition has defined the range of values of a partitioning attribute it can store. The third technique is *value-based*. In this technique, every partition has defined the set of values it can store. The fourth technique is based on the *round robin* algorithm.

Inserting, updating, and deleting data from partitioned tables are managed by a system. If a row is inserted, it is a system that select the right partition where the row will be stored. If a row is deleted, then the system finds a right partition where the row was stored. If the value of a partitioning attribute of an existing row is updated, then depending on a system and system parameters, a system may move an updated row from one to another partition.

In this chapter we focus on horizontal partitioning, which is natively supported by Oracle, DB2, and SQL Server. Vertical partitioning must be simulated in these three DB/DWMSs. Typically, vertical partitioning of table T is simulated by creating n separate tables T_i, each of which contains the subset of columns of T. Additionally, each of T_i must contain a primary key column(s) used for the reconstruction of the original table T. The reconstruction can be implemented by a view on top of tables T_i.

Vertical partitioning is supported in commercial and open source systems that use column storage, e.g., Sybase IQ, EMC Greenplum, C-Store/Vertica, MonetDB, Sadas, FastBit, Model 204.

2.5.1 Partitioning in Oracle

Oracle supports multiple partitioning techniques, namely: range, interval, list, hash, virtual column, system, reference, and composite. In this section, we will briefly overview the partitioning techniques.

Range Partitioning

In the range partitioning, each partition has defined its own ranges of values it can store. The ranges are applicable to a partitioning attribute. For example, the below table *Sales_Range_TKey* includes five partitions named *Sales_1Q_2009*, ..., *Sales_Others*. Partition *Sales_1Q_2009* accepts rows whose values of partitioning attribute *TimeKey* are lower than '01-04-2009'. Partition *Sales_2Q_2009* accepts rows whose values of the partitioning attribute fulfill the condition '01-04-2009' $\leq$ TimeKey $<$ '01-07-2009', etc. Notice, that in range partitioning partitions must be ordered by the ranges. Partition *Sales_Others* is defined with the `MAXVALUE` keyword. It allows to store all the other records having the value of *TimeKey* greater or equal to '01-01-2010'. Clause `tablespace` allows to point a tablespace where a partition is to be physically stored (an Oracle tablespace is a database object that allows to logically organize multiple files under one name).

```
create table Sales_Range_TKey
(ProductID varchar2(8) not null references Products(ProductID),
 TimeKey date not null references time(TimeKey),
 CustomerID varchar2(10) not null references Customers(CustomerID),
 SalesPrice number(6,2))
PARTITION by RANGE (TimeKey)
  (partition Sales_1Q_2009
      values less than (TO_DATE('01-04-2009', 'DD-MM-YYYY'))
         tablespace Data01,
   partition Sales_2Q_2009
      values less than (TO_DATE('01-07-2009', 'DD-MM-YYYY'))
         tablespace Data02,
   partition Sales_3Q_2009
      values less than (TO_DATE('01-10-2009', 'DD-MM-YYYY'))
         tablespace Data03,
   partition Sales_4Q_2009
      values less than (TO_DATE('01-01-2010', 'DD-MM-YYYY'))
         tablespace Data04,
   partition Sales_Others
      values less than (MAXVALUE) tablespace Data05);
```

Interval Partitioning

A special type of a range partition is an interval partition whose partitioning attribute is typically of type `date`. The difference between these two partitions is that interval partitions are automatically created by a system when needed. If data not fitting into existing partitions are to be inserted, then appropriate new partitions are created to store the data. Interval partitions are created by using the `interval` keyword followed by the definition of the interval. The interval is defined by means of a system function `NumToYMInterval` whose fist argument is the value (length) of an interval and the second one is its measurement unit. A fragment of an SQL command defining interval

partitioning is shown below. In this example, an interval is equal to three months.

```
...
PARTITION by RANGE (TimeKey)
INTERVAL (NumToYMInterval(3, 'MONTH'))
  (partition Sales_1Q_2009
     values less than (TO_DATE('01-04-2009', 'DD-MM-YYYY')),
   partition Sales_2Q_2009
     values less than (TO_DATE('01-07-2009', 'DD-MM-YYYY')));
```

List Partitioning

In the list partitioning each partition has assigned the set of values of a partitioning attribute that the partition accepts. The example below table *Sales_List_PayType* is divided into three partitions. Partition *Sales_Credit_Debit* stores sales record paid with a credit card ('Cr') or a debit card ('De'). Partition *Sales_Cash* stores sales records paid with cash. The last partition stores records having the value of *PaymentType* other than the three aforementioned. To this end, the `DEFAULT` keyword is used.

```
create table Sales_List_PayType
(ProductID varchar2(8) not null references Products(ProductID),
 TimeKey date not null references time(TimeKey),
 CustomerID varchar2(10) not null references Customers(CustomerID),
 SalesPrice number(6,2),
 PaymentType varchar(2))
PARTITION by LIST (PaymentType)
  (partition Sales_Credit_Debit values ('Cr','De') tablespace Data01,
   partition Sales_Cash values ('Ca') tablespace Data02,
   partition Sales_Others values (DEFAULT) tablespace Data05
);
```

Hash Partitioning

In the hash partitioning, data are placed in partitions by an internal Oracle hash function. As an example, let us consider table *Sales_Hash_CustID* that is composed of two partitions. Both of them have names assigned by a system and are stored in a default tablespace.

```
create table Sales_Hash_CustID
(ProductID varchar2(8) not null references Products(ProductID),
 TimeKey date not null references time(TimeKey),
 CustomerID varchar2(10) not null references Customers(CustomerID),
 SalesPrice number(6,2),
 PaymentType char(1))
PARTITION by HASH (CustomerID) partitions 2;
```

A user can explicitly assign names and storage locations for hash partitions. To this end, the `PARTITION by` clause has to be modified, as shown below.

```
...
PARTITION by HASH (CustomerID)
  (partition Cust1 tablespace Data01,
   partition Cust2 tablespace Data02));
```

Virtual Column Partitioning

Virtual column partitioning (available from Oracle11g) requires a virtual column in the definition of a table. A virtual column is a column whose value is computed either by a formula or a stored deterministic function (keyword `deterministic` in a function definition). Next, this column is used as a partitioning attribute, but when a virtual column is used as a partitioning attribute its values cannot be returned by a stored function. Typically, this type of partitioning is applicable either to range or list partitioning.

As an example, we show below a fragment of the command creating partitions based on virtual column *Gross*.

```
create table Products_Virt1
(...
 SellPrice number(6,2),
 Tax number(4,2),
 Gross as (SellPrice*Tax))
 PARTITION by range(Gross)
   (partition Prod1 values less than (1000),
    partition Prod2 values less than (2000));
```

Queries that use in their predicates either virtual column *Gross* or formula *SellPrice*Tax* can profit from the above partitioned table. For example, the simple query below

```
select * from Products_Virt1
where SellPrice*Tax<1000
```

will be answered by accessing only partition *Prod1*.

System Partitioning

In a system partitioning (available from Oracle11g), partitions do not have assigned any constraints and any row can be inserted into any partition. In this case, the DBMS does not control the placement of rows, i.e., it is a user (or application logic) that is responsible for inserting rows into the required partitions. In the system partitioning, every `insert` command must explicitly include the name of a partition where a row is to be inserted. A fragment of a command defining a system partitioned table is shown below.

```
create table Customers_Sys
(CustomerID varchar2(10) CONSTRAINT pk_Customers PRIMARY KEY,
```

```
 ...)
 PARTITION by SYSTEM
 (partition Cust_Europe, partition Cust_America, partition Cust_Asia);
```

Reference Partitioning

A reference partitioning (available from Oracle11g) is applicable to partitioning tables related to each other by primary key - foreign key relationships. A table with a primary key has defined explicitly a partition schema whereas a table with a foreign key inherits partitioning attribute and schema from the parent table. As an example illustrating this type of partitioning let us consider range partitioned table *Products_List_Cat*, as shown below.

```
create table Products_List_Cat
(ProductID varchar2(8) PRIMARY KEY,
 ProdName  varchar2(30),
 Category  varchar2(15),
 SellPrice number (6,2),
 Manufacturer varchar2(20))
 PARTITION by LIST(Category)
   (partition Prod_Elect values ('electronic'),
    partition Prod_Clo values ('clothes'));
```

Table *Sales_List_Cat* will inherit partitioning attribute and partition definitions from *Products_List_Cat*.

```
create table Sales_List_PayType
(ProductID varchar2(8) not null
      constraint ProdID_FK references Products_List_Cat(ProductID),
 ...)
 PARTITION by REFERENCE (ProdID_FK);
```

When tables are partitioned by reference, a query optimizer joins the tables by the *partition wise join*. In this join, only those partitions are joined that produce non empty set.

Composite Partitioning

Composite partitioning allows to divide main partitions into subpartitions. In Oracle11g main partitions can be either range or list, whereas subpartitions can be range, list, or hash.

```
create table Sales_Comp_RH
(ProductID varchar2(8) not null references Products(ProductID),
 TimeKey date not null references time(TimeKey),
 CustomerID varchar2(10) not null references Customers(CustomerID),
 SalesPrice number(6,2))
PARTITION by RANGE (TimeKey)
SUBPARTITION by HASH (ProductID) subpartitions 2
  (partition Sales_1Q_2009
```

```
    values less than (TO_DATE('01-04-2009', 'DD-MM-YYYY'))
       tablespace Users,
  partition Sales_2Q_2009
    values less than (TO_DATE('01-07-2009', 'DD-MM-YYYY'))
       tablespace Users,
  partition Sales_3Q_2009
    values less than (TO_DATE('01-10-2009', 'DD-MM-YYYY'))
       tablespace Users,
  partition Sales_4Q_2009
    values less than (TO_DATE('01-01-2010', 'DD-MM-YYYY'))
       tablespace Users);
```

DDL and Select on Partitioned Tables

When the value of a partitioning attribute of a given row is updated and this row no longer qualifies for its partition then the row can be automatically migrated into another partition. Automatic migration is available when for a partitioned table the `alter table TableName enable row movement` command is executed. Otherwise, such an update will not be possible.

Partitions can be explicitly addressed by queries, delete commands, and insert commands (the latter is possible only for system partitions). To this end, the table name must be followed by keyword `partition(PartitionName)`. If however, a query does not address a partition explicitly, a query optimizer, based on partition definitions and query predicates, will optimize the query and will address only these partitions that contribute to the query result.

2.5.2 Partitioning in DB2

DB2 supports range partitioning of tables and indexes. The mechanism is similar to the one described for Oracle. An example fragment of a command creating a range-partitioned table is shown below. For every partition, the range of values of a partitioning attribute (i.e., *TimeKey*) is defined, similarly as in Oracle. The below table is composed of four partitions, each of which stores sales from one quarter of year 2009.

```
create table Sales_Range_TKey
(iProductID varchar2(8) , ...)
PARTITION BY RANGE(TimeKey)
(partition Sales_1Q_2009 starting '01-01-2009',
 partition Sales_2Q_2009 starting '01-04-2009',
 partition Sales_3Q_2009 starting '01-07-2009',
 partition Sales_4Q_2009 starting '01-10-2009' ending '31-12-2009')
```

2.5.3 Partitioning in SQL Server

SQL Server (from version 2005) provides horizontal range partitioning, similar to Oracle and DB2. Partitioning applies to tables in indexes.

Partitioning is defined with the support of two database object, namely a partition function and a partition scheme. The *partition function* defines the number of partitions for a table and ranges of values for every partition. The *partition scheme* defines storage locations for table partitions. The definition of a partition scheme is based on the partition function.

An example partition function *Sales_Range_TKey*, defined for attribute of type `datetime`, is shown below. Applying this function to partitioning a table will result in five partitions having the following ranges of dates: date<'2009-04-01', '2009-04-01'≤ date <'2009-07-01', '2009-07-01'≤ date <'2009-10-01', '2009-10-01' ≤ date <'2010-01-01', and date≥'2010-01-01'. The `range right` or `range left` keywords define the policy of inclusion of border values. If the `range left` keyword is specified then the ranges for partitions would be defined as follows: date≤'2009-04-01', '2009-04-01'< date ≤ '2009-07-01', ..., and date>'2010-01-01'.

```
create PARTITION FUNCTION [Sales_Range_TKey] (datetime)
as RANGE right for values
  ('20090401', '20090701', '20091001', '20100101');
```

A partition function may be used also for numeric and character columns in the same way as illustrated above.

An example partition scheme *PS_Sales_Range_TKey* based on the *Sales_Range_TKey* partition function is shown below. Each of the five partitions created by the partition function is placed in a separate filegroup, whose name is given in the `to` clause. Partition storing range date<'2009-04-01' is stored in file group *Data01*, partition storing range '2009-04-01'≤ date <'2009-07-01' is stored in *Data02*, etc. A SQL Server filegroup is a database object offering the mechanism similar to a *tablespace* in Oracle, i.e., it allows to combine multiple files under a given name.

```
create PARTITION SCHEME PS_Sales_Range_TKey
as partition Sales_Range_TKey
to (Data01, Data02, Data03, Data04, Data05);
```

Finally, based on the defined partitioning scheme, a partitioned table can be created, as shown below.

```
create table Sales_Range_TKey
(ProductID varchar(8),
 TimeKey datetime ...)
on PS_Sales_Range_TKey (TimeKey)
```

2.6 Summary

Multiple research and technological works in the area of data warehouses have been focusing on providing means for increasing the performance of a data warehouse for analytical queries and other types of data processing. As mentioned already, DW performance depends on multiple components of a DW

architecture that include among others: hardware and computational architectures, physical storage schemes of data, query optimization and execution, dedicated data structures supporting faster data access.

In this chapter we focused on just one component from the aforementioned list, i.e., on data structures. The most popular data structures used in practice in major commercial DWMSs include: various index structures, materialized views, as well as partitioning of tables and indexes. In this chapter, first we discussed basic index structures, including a bitmap index, a join index, and a bitmap join index. We outlined the functionality of explicitly created bitmap indexes and bitmap join indexes in Oracle and we showed how they are applied in a query execution process. We also showed how system-managed bitmap indexes are applied to star query executions in DB2 and SQL Server. Second, we discussed the concept of materialized views and their application to query rewriting. We analyzed the functionality of materialized views in Oracle, DB2, and SQL Server. We showed how star queries are rewritten based on Oracle materialized views. Third, we discussed table partitioning techniques in Oracle, DB2, and SQL Server.

From the research and technological point of view, open issues in the area of DW performance include among others: building DWs in a parallel computation environments (cloud, grid, clusters, GPUs), main memory DWs, efficient data storage schemes (the column store and storage in MOLAP servers), query optimization and processing (especially in parallel computation environments and in main memory architectures), novel data structures supporting faster access to data, data and index compression techniques, testing and assessing a DW performance.

New business domains of DW application require more advanced DW functionalities, often combining the transactional and analytical features [82]. For example, monitoring unauthorized credit card usage, monitoring telecommunication networks and predicting their failures, monitoring car traffic, analyzing and predicting share rates, require accurate and up to date analytical reports. In order to fulfill this demand, one has to assure that the content of a DW is synchronized with the content of data sources with a minimum delay, e.g., seconds or minutes, rather than hours. Moreover, queries have to be answered instantly analyzing new data that were just loaded into a DW. To this end, the technology of a real-time (near real-time, right-time) data warehouse (RTDW) has been developed, e.g., [83, 84]. Strong demand for up to date data at any moment opens a new areas of research on assuring high performance of RTDWs.

Acknowledgements. This chapter was prepared during a research visit at Universidad de Costa Rica (San Jose, Costa Rica), supported from the project *Engineer's era. Expansion potential of Poznań University of Technology* (financed by the European Union as part of the Human Capital Operational Programme).

References

1. d'Orazio, L., Bimonte, S.: Multidimensional Arrays for Warehousing Data on Clouds. In: Hameurlain, A., Morvan, F., Tjoa, A.M. (eds.) Globe 2010. LNCS, vol. 6265, pp. 26–37. Springer, Heidelberg (2010)
2. Furtado, P.: A survey of parallel and distributed data warehouses. International Journal of Data Warehousing and Mining 5(2), 57–77 (2009)
3. Jhingran, A., Jou, S., Lee, W., Pham, T., Saha, B.: IBM Business Analytics and Cloud Computing: Best Practices for Deploying Cognos Business Intelligence to the IBM Cloud. MC Press, LLC (2010)
4. Andrzejewski, W., Wrembel, R.: GPU-WAH: Applying Gpus to Compressing Bitmap Indexes with Word Aligned Hybrid. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6262, pp. 315–329. Springer, Heidelberg (2010)
5. Fang, W., He, B., Luo, Q.: Database compression on graphics processors. Proc. VLDB Endow. 3, 670–680 (2010)
6. Govindaraju, N., Gray, J., Kumar, R., Manocha, D.: Gputerasort: high performance graphics co-processor sorting for large database management. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 325–336 (2006)
7. Govindaraju, N.K., Lloyd, B., Wang, W., Lin, M., Manocha, D.: Fast computation of database operations using graphics processors. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 215–226 (2004)
8. Lauer, T., Datta, A., Khadikov, Z., Anselm, C.: Exploring graphics processing units as parallel coprocessors for online aggregation. In: DOLAP, pp. 77–84 (2010)
9. Kemper, A., Neumann, T.: Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. In: ICDE, pp. 195–206 (2011)
10. Qiao, L., Raman, V., Reiss, F., Haas, P.J., Lohman, G.M.: Main-memory scan sharing for multi-core cpus. Proc. VLDB Endow. 1, 610–621 (2008)
11. Ross, K.A., Zaman, K.A.: Serving datacube tuples from main memory. In: Proc. of Int. Conf. on Scientific and Statistical Database Management (SSDBM), p. 182 (2000)
12. Abadi, D., Madden, S., Ferreira, M.: Integrating compression and execution in column-oriented database systems. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 671–682 (2006)
13. Abadi, D.J., Madden, S.R., Hachem, N.: Column-stores vs. row-stores: how different are they really? In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 967–980 (2008)
14. Stonebraker, M., Abadi, D.J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., O'Neil, P., Rasin, A., Tran, N., Zdonik, S.: C-store: a column-oriented dbms. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 553–564 (2005)
15. Abadi, D.J., Boncz, P.A., Harizopoulos, S.: Column-oriented database systems. Proc. VLDB Endow. 2, 1664–1665 (2009)
16. Cuzzocrea, A.: Data cube compression techniques: A theoretical review. In: Encyclopedia of Data Warehousing and Mining. IGI Global (2009)
17. Hasan, K.M.A., Tsuji, T., Higuchi, K.: An Efficient Implementation for MOLAP Basic Data Structure and its Evaluation. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 288–299. Springer, Heidelberg (2007)

18. Moerkotte, G.: Building query compilers, http://pi3.informatik.uni-mannheim.de/~moer/querycompiler.pdf (retrieved October 6, 2011)
19. Graefe, G.: A generalized join algorithm. In: BTW, pp. 267–286 (2011)
20. Graefe, G.: Parallel query execution algorithms. In: Encyclopedia of Database Systems, pp. 2030–2035. Springer, Heidelberg (2009)
21. Zeller, H., Graefe, G.: Parallel query optimization. In: Encyclopedia of Database Systems, pp. 2035–2038. Springer, Heidelberg (2009)
22. Chaudhuri, S.: Query optimizers: time to rethink the contract? In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 961–968 (2009)
23. Ioannidis, Y.E., Poosala, V.: Histogram-based approximation of set-valued query-answers. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 174–185 (1999)
24. Vitter, J.S., Wang, M.: Approximate computation of multidimensional aggregates of sparse data using wavelets. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 193–204 (1999)
25. Gramacki, A., Gramacki, J., Andrzejewski, W.: Probability density functions for calculating approximate aggregates. Foundations of Computing and Decision Sciences Journal 35 (2010)
26. Valduriez, P.: Join indices. ACM Transactions on Database Systems (TODS) 12(2), 218–246 (1987)
27. O'Neil, P.: Model 204 architecture and performance. In: Gawlick, D., Reuter, A., Haynie, M. (eds.) HPTS 1987. LNCS, vol. 359, pp. 40–59. Springer, Heidelberg (1989)
28. Stockinger, K., Wu, K.: Bitmap indices for data warehouses. In: Wrembel, R., Koncilia, C. (eds.) Data Warehouses and OLAP: Concepts, Architectures and Solutions, pp. 157–178. Idea Group Inc. (2007); ISBN 1-59904-364-5
29. Bryla, B., Loney, K.: Oracle Database 11g DBA Handbook. McGraw-Hill Osborne Media (2007)
30. O'Neil, P., Graefe, G.: Multi-table joins through bitmapped join indices. SIGMOD Record 24(3), 8–11 (1995)
31. Gupta, A., Mumick, I.S. (eds.): Materialized Views: Techniques, Implementations, and Applications. MIT Press (1999)
32. Furtado, P.: Workload-Based Placement and Join Processing in Node-Partitioned Data Warehouses. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 38–47. Springer, Heidelberg (2004)
33. Rao, J., Zhang, C., Megiddo, N., Lohman, G.: Automating physical database design in a parallel database. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 558–569 (2002)
34. Stöhr, T., Rahm, E.: Warlock: A data allocation tool for parallel warehouses. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 721–722 (2001)
35. Deliège, F., Pedersen, T.B.: Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. In: EDBT, pp. 228–239. ACM (2010)
36. Nourani, M., Tehranipour, M.H.: Rl-huffman encoding for test compression and power reduction in scan applications. ACM Trans. Design Autom. Electr. Syst. 10(1), 91–115 (2005)
37. Stabno, M., Wrembel, R.: RLH: Bitmap compression technique based on run-length and Huffman encoding. Information Systems 34(4-5), 400–414 (2009)
38. Aouiche, K., Darmont, J.: Data mining-based materialized view and index selection in data warehouses. J. Intell. Inf. Syst. 33, 65–93 (2009)

39. Lawrence, M., Rau-Chaplin, A.: Dynamic view selection for olap. In: Strategic Advancements in Utilizing Data Mining and Warehousing Technologies, pp. 91–106. IGI Global (2010)
40. Theodoratos, D., Xu, W., Simitsis, A.: Materialized view selection for data warehouse design. In: Encyclopedia of Data Warehousing and Mining, pp. 1182–1187. IGI Global (2009)
41. Chen, Y., Dehne, F.K.H.A., Eavis, T., Rau-Chaplin, A.: Improved data partitioning for building large rolap data cubes in parallel. IJDWM 2(1), 1–26 (2006)
42. Furtado, P.: Large Relations in Node-Partitioned Data Warehouses. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 555–560. Springer, Heidelberg (2005)
43. Binnig, C., Kossmann, D., Kraska, T., Loesing, S.: How is the weather tomorrow?: towards a benchmark for the cloud. In: Proc. of Int. Workshop on Testing Database Systems, DBTest (2009)
44. Funke, F., Kemper, A., Krompass, S., Neumann, T., Seibold, M., Kuno, H., Nica, A., Poess, M.: Metrics for measuring the performance of the mixed workload ch-benchmark. In: Proc. of Technology Conference on Performance Evaluation and Benchmarking, TPCTC (2011)
45. Kersten, M.L., Kemper, A., Markl, V., Nica, A., Poess, M., Sattler, K.U.: Tractor pulling on data warehouses. In: Proc. of Int. Workshop on Testing Database Systems, DBTest (2011)
46. O'Neil, P.E., O'Neil, E.J., Chen, X., Revilak, S.: The Star Schema Benchmark and Augmented Fact Table Indexing. In: Nambiar, R., Poess, M. (eds.) TPCTC 2009. LNCS, vol. 5895, pp. 237–252. Springer, Heidelberg (2009)
47. Gyssens, M., Lakshmanan, L.V.S.: A foundation for multi-dimensional databases. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 106–115 (1997)
48. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. Springer, Heidelberg (2003)
49. Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications. Springer Publishing Company, Inc., Heidelberg (2008)
50. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Record 26(1), 65–74 (1997)
51. Li, Z., Ross, K.A.: Fast joins using join indices. VLDB Journal 8(1), 1–24 (1999)
52. Davis, K.C., Gupta, A.: Indexing in data warehouses: Bitmaps and beyond. In: Wrembel, R., Koncilia, C. (eds.) Data Warehouses and OLAP: Concepts, Architectures and Solutions, pp. 179–202. Idea Group Inc. (2007); ISBN 1-59904-364-5
53. Wu, K., Otoo, E.J., Shoshani, A.: On the performance of bitmap indices for high cardinality attributes. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 24–35 (2004)
54. Scientific Data Management Research Group: FastBit: An efficient compressed bitmap index technology, `http://sdm.lbl.gov/fastbit/` (retrieved October 24, 2011)
55. Wu, M., Buchmann, A.: Encoded bitmap indexing for data warehouses. In: Proc. of Int. Conf. on Data Engineering (ICDE), pp. 220–230 (1998)
56. Wu, K., Yu, P.: Range-based bitmap indexing for high cardinality attributes with skew. In: Int. Computer Software and Applications Conference (COMPSAC), pp. 61–67 (1998)

57. Rotem, D., Stockinger, K., Wu, K.: Optimizing candidate check costs for bitmap indices. In: Proc. of ACM Conf. on Information and Knowledge Management (CIKM), pp. 648–655 (2005)
58. Rotem, D., Stockinger, K., Wu, K.: Optimizing I/O Costs of Multi-Dimensional Queries using Bitmap Indices. In: Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 220–229. Springer, Heidelberg (2005)
59. Stockinger, K., Wu, K., Shoshani, A.: Evaluation Strategies for Bitmap Indices with Binning. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 120–129. Springer, Heidelberg (2004)
60. Koudas, N.: Space efficient bitmap indexing. In: Proc. of ACM Conf. on Information and Knowledge Management (CIKM), pp. 194–201 (2000)
61. Chan, C., Ioannidis, Y.: Bitmap index design and evaluation. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 355–366 (1998)
62. O'Neil, P., Quass, D.: Improved query performance with variant indexes. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 38–49 (1997)
63. Rinfret, D., O'Neil, P., O'Neil, E.: Bit-sliced index arithmetic. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 47–57 (2001)
64. Antoshenkov, G., Ziauddin, M.: Query processing and optimization in Oracle RDB. VLDB Journal 5(4), 229–237 (1996)
65. Wu, K., Otoo, E.J., Shoshani, A.: An efficient compression scheme for bitmap indices. Research report, Lawrence Berkeley National Laboratory (2004)
66. Wu, K., Otoo, E.J., Shoshani, A.: Optimizing bitmap indices with efficient compression. ACM Transactions on Database Systems (TODS) 31(1), 1–38 (2006)
67. Deliège, F.: Concepts and Techniques for Flexible and Effective Music Data Management. PhD thesis, Aalborg University, Denmark (2009)
68. Stabno, M., Wrembel, R.: RLH: Bitmap compression technique based on run-length and Huffman encoding. In: Proc. of ACM Int. Workshop on Data Warehousing and OLAP (DOLAP), pp. 41–48 (2007)
69. Huffman, D.A.: A method for the construction of minimum-redundancy codes. In: Proc. of the Institute of Radio Engineers, pp. 1098–1101 (1952)
70. O'Neil, E., O'Neil, P., Wu, K.: Bitmap index design choices and their performance implications. Research report, Lawrence Berkeley National Laboratory (2007)
71. Reiss, F., Stockinger, K., Wu, K., Shoshani, A., Hellerstein, J.M.: Efficient analysis of live and historical streaming data and its application to cybersecurity. Research report, Lawrence Berkeley National Laboratory (2006)
72. Oracle Corp.: Oracle Database Data Warehousing Guide, Rel. 11g, `http://www.oracle.com/technetwork/database/enterprise-edition/documentation/database-093888.html` (retrieved June 24, 2010)
73. Bhattacharjee, B., Padmanabhan, S., Malkemus, T., Lai, T., Cranston, L., Huras, M.: Efficient query processing for multi-dimensionally clustered tables in db2. In: VLDB, pp. 963–974 (2003)
74. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13, 422–426 (1970)
75. Microsoft Corp.: SQL Server (2008) R2, `http://msdn.microsoft.com/enus/library/ms191267.aspx` (retrieved June 24, 2010)
76. Aouiche, K., Darmont, J., Boussaïd, O., Bentayeb, F.: Automatic Selection of Bitmap Join Indexes in Data Warehouse. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2005. LNCS, vol. 3589, pp. 64–73. Springer, Heidelberg (2005)

77. Hobbs, L., Hillson, S., Lawande, S.: Oracle9iR2 Data Warehousing. Digital Press (2003)
78. de Sousa, M.F., Sampaio, M.C.: Efficient materialization and use of views in data warehouses. SIGMOD Record 28(1), 78–83 (1999)
79. Gupta, H.: Selection of Views to Materialise in a Data Warehouse. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 98–112. Springer, Heidelberg (1996)
80. Lawrence, M., Rau-Chaplin, A.: Dynamic View Selection for OLAP. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 33–44. Springer, Heidelberg (2006)
81. Theodoratos, D., Xu, W.: Constructing search space for materialized view selection. In: Proc. of ACM Int. Workshop on Data Warehousing and OLAP (DOLAP), pp. 48–57 (2004)
82. Krueger, J., Tinnefeld, C., Grund, M., Zeier, A., Plattner, H.: A case for online mixed workload processing. In: Proc. of Int. Workshop on Testing Database Systems (DBTest). ACM (2010)
83. Castellanos, M., Dayal, U., Miller, R.J.: Enabling Real-Time Business Intelligence. LNBIP, vol. 41. Springer, Heidelberg (2010)
84. Thiele, M., Fischer, U., Lehner, W.: Partition-based workload scheduling in living data warehouse environments. Information Systems 34(4-5), 382–399 (2009)

3

OLAP Query Personalisation and Recommendation: An Introduction

Patrick Marcel

Laboratoire d'Informatique
Université François Rabelais Tours
France
patrick.marcel@univ-tours.fr

Summary. The aim of this lecture is to present how popular user-centric techniques, namely personalisation and recommendation, can be adapted to an OLAP context. The presentation begins with an overview of query personalisation and query recommendation in relational databases. Then it introduces the approaches proposed for personalising OLAP queries with user preferences, and the approaches proposed for recommending OLAP queries. All the approaches are characterized in terms of formulation effort, prescriptiveness, proactiveness, expressiveness, and in terms of the data leveraged: the current state of the database, its history, or external information.

Keywords: OLAP queries, preferences, query personalisation, collaborative filtering, recommender systems, query recommendation.

3.1 Introduction

According to a recent article of The Economist, mankind created 150 exabytes (billion gigabytes) of data in 2005. In 2010, it will create 1,200 exabytes [1]. With such amounts of data, it is of paramount importance to be able to access relevant information efficiently, using sophisticated search tools. On the other hand, systems offering search facilities, like DBMSs, are quite uneasy to use, driving querying or navigation into huge amount of data a very tedious process. Those such facilities should be more user-friendly [2].

In domains like the Web, Information Retrieval or e-commerce, user-centric approaches like personalisation or recommendation have been proved successful (see e.g., [3]). It is for instance believed that Amazon makes around 30% of sales thanks to recommendations. Such approaches are very relevant in a database context. For instance, the user may not accept to spend too much time conceiving the query. In addition, she may not be happy if the query's answer shows too many or too few results. And, even if the size of the answer is acceptable, she may be relieved to see the system automatically suggesting

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 63–83, 2012.

queries that will display other answers of interest, especially if she is left with the task of navigating the database to analyse the data it contains.

A typical example of such an analysis is that of a data warehouse navigated by decision makers using OLAP queries [4]. A data warehouse can be seen as a large database with a particular topology, shared by many analysts who have various interests and viewpoints, where data is seen as a cube, explored by sequences of OLAP queries that may return large answers. In such a context, being able to personalize or recommend queries is seen as particularly relevant [5].

This lecture introduces how personalisation and recommendation approaches have been adopted by the database community, and how they can be adapted to these particular databases that are data warehouses. More precisely, it will try to answer the following two questions:

- Given a database query q, how to to cope with too many/too few results? Personalisation can be used to answer this question. If the query result is too large then being able to add preferences to this query gives a way of ranking the query results to focus on the most relevant first. On the other hand, if the query result is too small, then selection predicates, also called strong constraints, could be turned into preferences (or soft constraints) to weaken this query.
- Given a sequence of queries over a database, how to suggest queries to pursue the session? In this case, what the user did in the past, or alternatively, what similar users did in the past, can serve as a basis for recommending relevant queries to complement the current query answer.

The lecture is organised as follows. The next section presents basic definitions and concepts of preferences and recommender systems. Section 3.3 delimits the scope of this lecture and precise how the approaches surveyed will be categorised. Section 3.4 and Section 3.5 introduce the existing approaches in relational databases and multidimensional databases, respectively. Section 3.6 concludes the lecture with a brief discussion and presents some open issues.

Note that this lecture assumes basic knowledge of relational database [6] and data warehousing [7]. Part of the material that focuses on query recommendation is borrowed from [8]. The slides illustrating this lecture are available on the author's web page [9].

3.2 Preliminaries

In this section, we provide the basic definitions underlying personalisation and recommendation.

3.2.1 Preference Expression

We begin by explaining the basics of preference modelling. A more comprehensive introduction can be found in [10].

Qualitative and Quantitative Preferences

Two types of approaches are used to express preferences. Qualitative approaches express relative preferences i.e., "I like a better than b". Such a preference is noted $a > b$ where $>$ is usually a Strict Partial Order (SPO). An SPO is a binary relation $>$ over a set O which is

- Irreflexive, i.e., for all $a \in O$, $\neg(a > a)$
- Asymmetric, i.e., for all $a, b \in O$, if $(a \neq b)$ and $(a > b)$ then $\neg(b > a)$
- Transitive, i.e., for all $a, b, c \in O$, if $(a > b)$ and $(b > c)$ then $(a > c)$

Given a preference relation $>$, the indifference relation $\sim$ is defined by: $(a \sim b)$ if $\neg(a > b)$ and $\neg(b > a)$. It expresses that a and b are not comparable. Particular partial orders of interest are Total Orders (TO) and Weak Orders (WO). A relation $>$ is a TO if for every a and b, either $(a > b)$ or $(b > a)$. $>$ is a WO if $>$ is a SPO and $\sim$ is transitive.

Example 1. Consider the following database instance:

Movies	Author	Genre	Price	Duration
t1	Cohen	Comedy	5	90
t2	Cohen	Comedy	6	100
t3	Cohen	Comedy	7	80
t4	Allen	Drama	7	120
t5	Lynch	Drama	5	150

The preference "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies" entails that tuple t5 is preferred to tuple t4 and tuple t4 is preferred to tuples t1, t2, t3. As preferences are assumed to be transitive, we say that t5 dominates all the other tuples. Note that this preference says nothing e.g., for t1 and t2, neither for t1 and t3.

Quantitative approaches express absolute preferences, i.e., I (do not) like a to a specific degree. They are based on Scoring / Utility Functions. I like a better than b is noted $u(a) > u(b)$ where u is a scoring function.

Quantitative approaches are often said to be less general than qualitative approaches in the sense that, in order to be representable using scoring functions, a preference relation has to be a WO, which implies that the corresponding indifference relation has to be transitive. But on the other hand, only scoring functions can express the intensity of preferences.

Example 2. The preference "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies" can be expressed by the following scoring function (assuming that scores range from 0 to 1):

- "I like Lynch" corresponds to a score of 0.9
- "I like Allen" corresponds to a score of 0.8
- "I like Cohen" corresponds to a score of 0.5

It can easily be seen that for instance, preference "I prefer cheaper movies, given that author and genre are the same" (i.e., t1 is preferred to both t2 and t3, but it is not preferred to t4 or t5) cannot be expressed with scoring functions.

Preference Composition

Preferences can be defined extensionally under the form relation instances, or intentionally. In the latter case, the intentional definition is called a model of preference. Models of preferences can be expressed with a given language like those of [11, 12] and/or by using composition of preference relations. Compositions follow the approach used to express preferences and thus can be qualitative or quantitative.

Qualitative Composition

In what follows, let T be a set of tuples and $>_1$ and $>_2$ be two preference relations over T. We restrict here to single dimensional composition, where preferences are expressed over a single relation. Common single dimensional composition includes Boolean Composition, Prioritized Composition, and Pareto Composition.

Boolean composition involves a boolean operator, for instance:

- Intersection, i.e., $R = (>_1 \cap >_2)$ with $(t\ R\ t')$ if $(t >_1 t')$ and $(t >_2 t')$
- Union, i.e., $R = (>_1 \cup >_2)$ with $(t\ R\ t')$ if $(t >_1 t')$ or $(t >_2 t')$

Prioritized Composition imposes a priority of a preference over another. It is formally defined by $R = (>_1 \lhd >_2)$ with $(t\ R\ t')$ if $(t >_1 t')$ or $(\neg(t' >_1 t)$ and $(t >_2 t'))$

Pareto Composition assumes two preferences to be equally important. It is formally defined by: $R = (>_1 \otimes >_2)$ with $(t\ R\ t')$ if $((t >_1 t')$ and $(t >_2 t'$ or $t \sim_2 t'))$ or $((t >_2 t')$ and $(t >_1 t'$ or $t \sim_1 t'))$.

Example 3. Consider the two preferences: "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies", called $P1$ and "I also prefer shorter movies" called $P2$. Composing them using intersection result in a particular SPO with no domination, that can be interpreted as everything is preferred since all tuples are undominated. Composing them with union violates irreflexivity and asymmetry, and thus the resulting relation is usually considered as not being a preference relation. Indeed, in this case, the resulting relation would mean for instance that both t5 should be preferred to t4 (according to $P1$) and t4 should be preferred to t5 (according to $P2$). Composing them with prioritisation results in a total order reflecting $P1$

first and then $P2$ only when $P2$ does not contradict $P1$. More precisely, we have t5 preferred to t4 preferred to t3 preferred to t1 preferred to t2. Finally, composing them using Pareto results in a preference relation where only t3 dominates t1 and t1 dominates t2, and neither t4 dominates t5 nor t5 dominates t4 since $P1$ and $P2$ do not agree for these two tuples.

Note that properties preservation (irreflexivity, etc.) under various kind of composition operators has been deeply studied (see [10] for more details).

Quantitative Composition

Quantitative composition is generally achieved through dedicated functions, like weighted functions, min, max, etc. An example of quantitative composition is, given preferences $P1$ modelled with $score_{P1}$ and preference $P2$ modelled with $score_{P2}$: $Score_{f(P1,P2)}(ti) = x \times score_{P1}(ti) + (1-x) \times score_{P2}(ti)$ where x is some weight.

Example 4. Consider the following preferences: "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies" (P1) expressed by:

- "I like Lynch" with $score_{P1} = 0.9$
- "I like Allen" with $score_{P1} = 0.8$
- "I like Cohen" with $score_{P1} = 0.5$

and "I also prefer shorter movies" (P2) expressed by:

- "I like (duration=80)" with $score_{P2} = 1$
- "I like (duration=90)" with $score_{P2} = 0.9$
- "I like (duration=150)" with $score_{P2} = 0.6$

Suppose we use the scoring function defined above to compose P1 and P2, with $x = 0.5$. Then, for instance, the score of tuple $t1$ would be: $0.5 \times 0.5 + 0.5 \times 0.9 = 0.7$

3.2.2 Recommender Systems

In this section, we briefly introduce the basics of recommender systems (see [13] for a more substantial presentation).

Basic Formulation

A recommender system is typically modelled as follows. Let I be a set of items (e.g., products in a typical e-commerce application) and U be a set of users (e.g., customers in a typical e-commerce application). Let f be an utility function with signature $U \times I \rightarrow R$ for some totally ordered set R (typically reals between 0 and 1). Recommending s' to u is to choose for the user u the item s' that maximizes the user's utility, i.e., $s' = argmax_I f(u, i)$. The function f can be represented as a matrix $M = U \times I$, that records for any user

u in U, any item i in I, the utility of i for u, that is $f(u, i)$. The problem of recommending items to users is that this matrix is both very large and very sparse. Thus, many methods have been proposed for estimating the missing ratings. Moreover, achieving relevant user-specific recommendations is particularly difficult since it has been observed that everyone is a bit eccentric [14].

In general, recommendation methods are categorized [13] into: (i) Content-based, that recommend items to the user u similar to previous items highly rated by u, (ii) Collaborative, that consider users similar (i.e. having similar profiles) to the one for which recommendations are to be computed as a basis for estimating its ratings and (iii) Hybrid, that combine content-based and collaborative ones. Note that [15] propose a multidimensional generalisation of this basic two-dimensional formulation, especially to support profiling and contextualisation.

Content-Based Recommendation

Typical content-based recommendation is based on the comparison between item profiles and user profiles. For instance, it can rely on the following principle:

1. Build item profiles by using selected features and providing a score for each feature.
2. Build user profiles from highly rated item profiles, typically by computing a weighted average of item profiles.
3. Compare user profiles with non-rated item profiles to estimate the missing ratings. Typical similarity measures include vector-based similarities like cosine.
4. Recommend to the user those non-rated items achieving the best similarity scores.

Example 5. Consider the following matrix recording ratings:

	Donuts	Duff	Apple	Tofu	Water	Bud	Ribs
Homer	0.9	0.8				0.7	
Marge			0.8		0.6		
Bart	0.7	0.6	0.1				0.8
Lisa	0.2			0.8	0.6		
Maggie	0.6			0.5	0.6		

Suppose that the features chosen for the profiles are: (contains sugar, ok for a diet). Item profiles are modelled as vectors recording a score for these two features. Suppose here that the scores are automatically computed from the items' nutrition facts. For instance, the profile for Donuts is $(0.9, 0)$ and the profile for Apple is $(0.4, 0.6)$. Then user profiles are also modelled as vectors recording scores for the same two features, derived from the known ratings.

For instance, Homer profile would be: $(0.9 \times (0.9, 0) + 0.8 \times (0.6, 0.1) + 0.7 \times (0.6, 0.1))/3 = (0.8, 0.1)$[1]. Lisa profile would be: $(0.3, 0.8)$. The similarity score for the user profile with the item profile estimates the missing ratings.

The limitations of content-based approaches are the following: First finding a good set of features must be done very carefully since it impacts directly the score estimates and hence the quality of the recommendation. Another problem is that recommendations stick to the user profile. For instance, using the example above, Homer will never be recommended Tofu. Finally, this approach suffers from the cold-start problem, i.e., how to build a profile for a new user for who no ratings are known.

Collaborative Recommendation

The main idea of collaborative approaches is to benefit from all users' ratings when estimating a user's missing ratings. Depending on how the matrix is used (row-wise or column-wise), two techniques are distinguished, that are based on computing similarities among users or items:

- User-user collaborative approaches estimate the ratings for items based on ratings of similar users.
- Item-item collaborative approaches estimate the ratings for items based on ratings for similar items.

Example 6. To illustrate the user-user approach, suppose that all user are modelled as vectors having as many components as there exists items, the value of the component being the rating for the item, or 0 if the rating is not known. For instance, Homer would be modelled as the following vector: (0.9,0.8,0,0,0,0.7,0). Similarity is computed between users, with cosine for instance, to find the users who are the most similar to the one for which the ratings are to be estimated. These users' ratings are derived to estimate the user's missing ratings. For instance, suppose that Bart is found the most similar to Homer, with a similarity score of 0.8. Then, given that Bart has a score for Ribs and Homer has not, Bart's score is used to estimate Homer's, by weighting Bart's score with the similarity between Bart and Homer, i.e., 0.8×0.8 is our example.

The limitations of the collaborative approach are that it relies on heavy pre-computation, and that new users or new items, for which no ratings are known, cannot be taken into account.

Hybrid Methods

Hybrid approaches are used to cope with the limitations of both previous approaches. Such approaches include the aggregation of a content-based

[1] This profile can be interpreted as: Homer contains sugar and is not ok for a diet.

computed score with a collaborative computed score, the addition of content-based to collaborative filtering, or the use of item profiles to cope with the new item problem.

3.3 Categorising the Approaches

In this section we define precisely the scope of the lecture and present the criteria used to categorise the approaches.

3.3.1 Scope of the Lecture

There is a lot of works in the database community that deal with query transformation, i.e., the process of, given a database query q, transforming q into another query q'. Among these work, we restrict our lecture to the following transformations:

- Query personalisation: given a database query q and some profile, compute a query $q' \subseteq q$ that has an added value w.r.t. the profile.
- Query recommendation: given a database query q and some profile, compute a query q' such that neither $q' \subseteq q$ nor $q \subseteq q'$, that has an added value w.r.t. the profile. Note that computing a query q' that include q would correspond to query relaxation (see e.g., [16, 17]).

Note that other forms of query transformation like e.g., relaxation, non relational data types (XML, etc.), and implementation and evaluation issues will not be covered.

3.3.2 Categorisation of the Approaches

To describe and categorise the approaches presented in this lecture, we adopt the criteria introduced in [18], that are mostly used to differentiate personalisation approaches.

- Formulation effort: some approaches require the user to manually specify profile elements for each query, while in others the best they are inferred from the context and the user past actions.
- Prescriptiveness: some approaches use profile elements as hard constraints that are added to a query while in other as soft ones: tuples that satisfy as much profile criteria as possible are returned even if no tuples satisfies all of them.
- Proactiveness: distinguishes the approaches that suggest new queries based on the navigation log and on the context (but that does not execute them), with respect to those that change the current query or post process its results before returning them to the user.
- Expressiveness: personalization criteria have different expressivenesses and can be differently combined.

To precisely distinguish between the type of data needed, especially for recommendation techniques, we also use the taxonomy proposed in [19]. There, three categories are identified:

- Current-state approaches, exploiting the content and schema of the current query result and database instance. Current-state approaches can be based either on (i) the local analysis of the properties of the result of the posed query or (ii) the global analysis of the properties of the database. In both cases systems exploit (i) the content and/or (ii) the schema of the query result or the database.
- History-based approaches, using the query logs.
- External sources approaches, i.e., approaches exploiting resources external to the database.

3.4 Query Personalisation and Recommendation in Relational Databases

In this section, we give a brief overview of the approaches developed in the relational database context.

3.4.1 Personalisation in Databases

We can distinguish two types of approaches:

- The use of explicit preference operators in queries. The most representative works in this category include Winnow [12], Preference SQL [11] and Skyline [20] This type of approaches requires high formulation effort, is not prescriptive not proactive, but is highly expressive.
- The rewriting (expansion) of regular database queries based on a profile. The most representative work is initiated in [21]. This approach requires a low formulation effort, is prescriptive and not proactive and has low expressiveness.

Use of Dedicated Operators

The basic definition of the operator computing dominating tuples is the following. Given a relation r with schema $sch(r)$ and a preference C over $sch(r)$ defining a preference relation $>_C$, the Winnow operator [12], denoted w_C, is defined by: $w_C(r) = \{t \in r | (\nexists t' \in r)(t' >_C t)\}$.

This operator can be used to order the query answer. Indeed, the answer to a query q can be partitioned according to C, i.e., $q = w_C(q) \cup w_C(q - w_C(q)) \cup \ldots$ meaning that the answer can be presented by displaying $w_C(q)$ first, then $w_C(q - w_C(q))$, etc.

Example 7. Suppose that preference C is "I prefer drama". The query "What are my most preferred affordable movies?" can be expressed by: $w_C(\sigma_{Price<7}(Movies))$. The answer can be presented by displaying t5 first, and then t1 and t2.

The work of Kiessling [11] extends this idea of having an operator dedicated to preference expression, to enrich the SQL syntax with a PREFERRING clause that enables the use of specific preference constructors. Each preference constructor can be use to express a specific atomic preference. Preferences can be composed with Pareto or prioritisation.

Example 8. Consider the following queries expressed with Preference SQL:

1. SELECT * FROM Movies PREFERRING HIGHEST(Duration)
2. SELECT * FROM Movies PREFERRING Genre IN ('Drama','Thriller')

The model of preference specified by the first query is the following: for some duration x and y, $(x >_{HIGHEST} y)$ if value x is greater than value y, meaning that movies with highest durations will be preferred. For the second query, the model of preference says that Drama and Thriller movies will be preferred over any other genre. More formally, if x and y are two movie genres, $(x >_{IN(`Drama',`Thriller')} y)$ if $x \in \{`Drama', `Thriller'\}$ and $y \notin \{`Drama', `Thriller'\}$.

A restricted form of Kiessling's SQL extension is the addition of the Skyline operator to SQL [20]. This is a restriction in the sense that, originally, Skyline queries feature only numerical attributes and Pareto composition. This operator is defined as follows. The syntax of a skyline clause is:

$$\begin{aligned}\text{SKYLINE OF } & d_1 \text{ MIN}, \ldots, d_k \text{ MIN} \\ & d_{k+1} \text{ MAX}, \ldots, d_l \text{ MAX} \\ & d_{l+1} \text{ DIFF}, \ldots, d_m \text{ DIFF}\end{aligned}$$

The semantics of such a clause is that a tuple $p = (p_1, \ldots, p_n)$ dominates a tuple $q = (q_1, \ldots, q_n)$ if:

$p_i \leq q_i$, for $i = 1, \ldots, k$
$p_i \geq q_i$, for $i = k+1, \ldots, l$
$p_i = q_i$, for $i = l+1, \ldots, m$

Query Expansion

In the absence of a dedicated preference operator, a regular user query can be processed and transformed with preferences, resulting in another regular query that is typically a subquery of the initial one. We use the work of [21] to illustrate this approach.

In this work, preferences come from a user profile which is modelled as a graph of atomic quantitative preferences of the form (selection condition, score), where selection is a regular selection predicate (that may be used to

join two tables) and score is a real between 0 and 1 indicating the intensity of the preference. Atomic preferences are composed using a very simple principle: composition of preference $(s1, v1)$ with preference $(s2, v2)$ results in preference $(s1 \wedge s2, v1 \times v2)$.

Query expansion is performed as follows. First, given a query, the k most relevant preferences are selected from the profile. Then the selected preferences are added as hard constraints to the query, and the query is finally executed.

Example 9. Consider the following user query: SELECT title FROM Movies WHERE duration $<$ 120. Suppose that the best preference selected from the profile of the user who wrote the query is "I like Lynch as Author". Then the query is modified, resulting in: SELECT title FROM Movies WHERE duration $<$ 120 AND Author='Lynch'. Note that in this example, if the query is evaluated over the instance given in Example 1, then the result is be empty.

This work has been extended to take into account constraints like result cardinality or execution time [22].

3.4.2 Query Recommendation in Databases

Recently, to our knowledge, there has been only two attempts to formalize database query recommendations for exploration purpose [23, 19]. It is important to see that given the context of database exploration, a direct transposition of the users $\times$ items matrix is not relevant. Following [19], we use the categories introduced Section 3.3 to categorise these approaches.

Current State

In [19], the authors focus on the current state approach and propose two techniques to recommend queries based on the database instance and/or the current query answer. The first technique, called local analysis, analyse the answer to the user query to discover patterns and use these patterns to recommend. The second technique, called global analysis, extends this principle to the entire database instance. The instance would have to be analysed off-line, for instance to discover correlations among attribute values.

Example 10. Consider the current query: SELECT Author, Genre FROM Movies WHERE Duration $>$ 100. Suppose that by analysing this query answer, it is found that the result has a lot of tuples whose genre is Drama. Then, a possible recommendation would be: SELECT Author, Genre FROM Movies WHERE Genre='Drama'. Suppose now that a global analysis of the database instance shows that value 'Cohen' for Author is correlated with value 'Comedy' for Genre. Then, if the current query is: SELECT * FROM Movies WHERE Author='Cohen' a recommendation would be: SELECT * FROM Movies WHERE Genre='Comedy'.

History Based

For [23], the problem of query recommendation is viewed as a sessions × tuples matrix. With this approach, a query is represented as a vector whose arity is the number of tuples of the database instance. The basic approach considers that each component of such a query is either a 1, if the query used the tuple, or a 0 otherwise. A session is also represented by a binary vector which is the logical or of the vectors of the queries of the session. Consider a particular session S_c called the current session. Recommendations for S_c are computed as follows. First, sessions similar to S_c are found, using some vector similarity measures like e.g., cosine. For those session closest to S_c, the queries they contain are also compared to S_c using the same similarity measure. Finally, the queries of those sessions that are the most similar to S_c are recommended.

In subsequent works [24], the authors focus on fragments (attributes, tables, joins and predicates) of queries and consider thus a sessions × query fragments matrix. In this work, the matrix is used column-wise in the sense that recommendation computation relies on fragment similarity instead of session similarity.

Query Completion

We conclude this section by noting that recommendation also make sense to assist the user writing a query. For instance, the SnipSuggest approach [25] is a collaborative approach that uses a query log to provide on-the-fly assistance to users writing complex queries. The query log is analysed to construct a graph whose nodes are the fragments appearing in queries and edges indicate the precedence. The edges are labelled with the probability that a fragment follows another fragment in the logged queries. Given the beginning of a current query, the graph is used to complete the query with the fragment that is the most likely to appear.

Example 11. Suppose that the query log contains only the following two queries: SELECT Title, Genre FROM Movies WHERE Actor='C. Lee' and SELECT Title FROM Movies WHERE Author='Allen'. Suppose that a user starts writing a query with only SELECT. It can be then suggested the attribute Title since this attribute is the most likely to appear according to the query log.

3.5 OLAP Query Personalisation and Recommendation in Data Warehouses

In this section, we introduce the personalisation and recommendation approaches that are tailored to data warehouses queries. We start by reviewing the salient peculiarities of data warehouses compared to classical relational databases.

3.5.1 Peculiarities of Data Warehouses

As evidenced by e.g., [26, 27, 28] basic peculiarities of typical data warehouse can be summarized by:

1. A data warehouse is a read-mostly database and its instance has an inflationist evolution (data are added, never or very seldom deleted). It is for instance likely that a user issues periodically the same sequence of queries more than once.
2. A data warehouse is a database shared by multiple users whose interests may vary over time. It is argued in [29, 28, 30, 5] that user preferences are of particular importance in data warehouse exploration. It would for instance be important to issue recommendations computed from other users' habits (e.g., in a collaborative filtering fashion) and at the same time respecting the user interests.
3. A data warehouse has a particular schema that reflects a known topology, often called the lattice of cuboids, which is systematically used for navigation [31]. Rollup and drilldown operations that allow to see facts at various levels of detail are very popular in this context.
4. A typical analysis session over a data warehouse is a sequence of queries having an analytical goal, each one written based on the past results of the session. They may be expressed in a dedicated query language (like e.g., MDX [32]), may produce large results that are usually visualised as crosstabs. Moreover, the session has a sense w.r.t. some expectations. For instance, the user may assume a uniform distribution of the data [33, 4] or that two populations follow the same distribution [34]. Sessions (as sequences of queries) are of particular importance in this context since by this sequence the user navigates to discover valuable insights w.r.t. her expectations or assumptions.

We now describe how these peculiarities have been taken into account when personalising or recommending OLAP queries.

3.5.2 Personalisation of OLAP Queries

To the best of our knowledge, only two works deal explicitly with the personalisation of OLAP queries. The first work [29] borrows from the query expansion paradigm, where a query expressed in MDX is transformed into another MDX query by using elements of the user profile. This approach features the same characteristics as that of [21] in terms of formulation effort, prescriptiveness, proactiveness and expressiveness. The second work [30, 35] is inspired by the use of explicit preference constructors enabling to express complex preferences directly within the query. This approach features the same characteristics as that of [11] in terms of formulation effort, prescriptiveness, proactiveness and expressiveness.

Query Expansion

The work of [29] is inspired by that of [22], the main difference being that it does not use scoring function to represent preferences but relies on a qualitative model instead. It proposes to expand a current MDX query by another MDX query q using those elements of the profile that guarantee that (i) q is included (in the classical sense of query inclusion) in the current query, (ii) q only fetches preferred facts w.r.t. the profile and (iii) q respects a visualisation constraint. In this work, the user profile is given by a qualitative preference model relying on orders defined over dimension names and over attribute values. The visualisation constraint is used to indicate the maximum number of references (i.e., positions extracted from a data cube) that can be used for displaying the query answer. Note that this work assumes that the instances of the dimension tables can be used to compute the most preferred references in terms of the user profile.

Example 12. Consider the following MDX query:

```
SELECT CROSSJOIN({City.Tours, City.Orleans},Category.Members)
       ON ROWS
       {2003, 2004, 2005, 2006} ON COLUMNS
FROM   SalesCube
WHERE (Measures.quantity)
```

This query asks for 2 cities, 5 members of the Category level and 4 years, i.e., 40 cells of the cube. Suppose that the device for displaying the query result imposes the use of a cross tab with 2 axes with only 4 positions, i.e., only 16 cells. Suppose also that the user profile states that the most recent years are preferred and that, for the product dimension, the preference are: Electronics $<$ shoes $<$ cloth $<$ food $<$ drink.

The personalisation process is as follows. First the most preferred references are computed. In this example, that would be references *(Orleans, 2006, drink, quantity)* and *(Tours, 2006, drink, quantity)*. Then it is checked if this set of references complies with the visualisation constraint. In this example, as there are 2 preferred references for 16 available positions, this is indeed the case. When it is the case, then the second preferred references are added to the first preferred and again the whole set is checked against the visualisation constraint. The process stops when no more positions are available in the cross tab used to display the result, or all the references requested by the query have been included in the cross tab. Then the set of references is used to construct the personalised query, which, in this example, would be:

```
SELECT CROSSJOIN(City.Tours, City.Orleans,Category.Food, Category.drink)
       ON ROWS
       2003, 2004, 2005, 2006 ON COLUMNS
FROM   SalesCube
WHERE (Measures.quantity)
```

Use of Dedicated Operators

The work of [30, 35] is inspired by that of [11] where preferences are written for each query using a dedicated PREFERRING clause added to the MDX language. In this work, preference constructors are tailored to the multidimensional context. For expressing atomic preferences, the main differences with [11] are thus the following:

- The semantics of preferences over attribute values is hierarchy-driven.
- Preferences can be expressed over levels and thus over cuboids.
- Preferences can be expressed over measures.

As in [11], atomic preferences can be composed using Prioritization or Pareto. A specific implementation has been developed for evaluating preference queries expressed in this language.

Example 13. We illustrate two representative preference constructors. Suppose that one of the hierarchies of the 5-dimensional schema is named RESIDENCE and has levels City, State, Country. The preference PREFERRING City IN 'LA' not only means that City 'LA' is preferred over all other cities, but also that ancestors and descendants of 'LA' are preferred over values of this hierarchy that are neither ancestors nor descendants of 'LA'. For instance, the reference *(LA, all, 2010, F, all)* will be preferred over, say, *(NY,all,all,all,all)*, and the reference *(California,all,2009,all,all)* will be preferred over e.g., *(NY,all,2010,all,all)*. Note that *(LA, all, 2010, F, all)* $\sim$ *(California,all,2009,all,all)*

As another example, PREFERRING RESIDENCE CONTAIN City means that the facts grouping by level City are preferred over the facts grouping by another level of the RESIDENCE hierarchy. In that case, reference *(LA,all,2010,F,all)* is preferred over *(California,all,2009,all,all)*.

3.5.3 Recommending OLAP Queries

Surprisingly, although there are many works in relational database dealing with query personalisation and quite a few works around query recommendations, in data warehouse, that seems to be the other way round. Indeed, to the best of our knowledge, the existing approaches for recommending OLAP queries are [36, 37, 38, 27, 33, 4, 39, 40, 41, 42, 43, 44, 45]. Note that the older works do not make use of the term query recommendation.

All these approaches are proactive, prescriptive, require a low formulation effort and have low expressiveness. To categorise them, we use and refine the categories proposed by [19]. We distinguish between (1) current-state methods exploiting external information (more precisely a user profile), (2) current-state methods based on expectations, (3) history-based methods exploiting query logs, and (4) hybrid (history and current-state) methods based on expectations.

Methods Exploiting a Profile

The works in this category [36, 37] suppose that a profile is provided together with the current query. The profile expresses user preferences over the tuples of the fact table using a quantitative approach. As in classical query expansion, the profile is used to modify the current query. The main difference with [21] is that the expanded query is not necessarily a subquery of the initial query.

Methods Based on Expectations

The works in this category [38, 27, 33, 4, 39] rely on discovery driven analysis, where a model on unseen data is used together with the already seen data, i.e., the results of the already launched queries of a given session. The strongest deviations to the model are recommended.

We briefly recall the concept of discovery driven analysis. To support interactive analysis of multidimensional data, [27] introduced discovery driven analysis of OLAP cubes as the definition of advanced OLAP operators to guide the user towards interesting regions of the cube, lightening the burden of a tedious navigation. These operators are of two kinds. The first kind tries to explain an unexpected significant difference observed in a query result by either looking for more detailed data contributing to the difference [33], or looking for less detailed data that confirm an observed tendency [39]. The second kind proposes to the user unexpected data in the cube w.r.t. the data she has already observed, by adapting the Maximum Entropy Principle [4].

Recommendation methods based on discovery driven analysis consist in recommending queries that result in data deviating the most from a consensual model (that we call expectation). Among the works in this approach, the main difference is the model used, i.e., the nature of the expectation. [33, 39, 4] rely on the assumption of a uniform data distribution. [38] assume a statistical independence of the cube's dimensions.

Example 14. We briefly illustrate the work of [4]. Suppose that a user asked for the total sale of Cheese in Europe for Quarter 1 of Year 2010. The result indicates that this sale is 100. Suppose that it is known (because the OLAP server transparently evaluates queries related to what the user is doing) that at the country level, this 100 is perfectly distributed among the 4 countries Europe drills down to (i.e., the sales of Cheese in these countries are 25 for each of France, Italy, Spain and U.K.). If such an uniform assumption is usually believed to be true, then this drill down does not correspond to a relevant recommendation. On the other hand, if at the month level, it is found that the sales of cheese in Europe is quite different from one month to another, then drilling down to the month level would be a good recommendation.

Note that [38] and [33, 39] compute suggestions using only the current query while [4] considers every former query result.

Methods Exploiting Query Logs

The works in this category [40, 41, 42, 43] suppose that a query log is used to look for similarities between the current session and former sessions, to extract one past query as the recommendation.

The methods in this category mainly differ by the way they consider the similarity between sessions and/or queries. [41] use the classical Levenshtein (for session) and Hausdorff (for queries) distances. [42, 43] group queries by common projections and selections, and use a Markov model to represent sessions. [40] cluster queries using the Hausdorff distance and detects if the current session is a prefix of some existing session. [42, 43] and [40] identify a matching position for the current session in the closest former session and recommend the query after this position. [41] recommend the last query of the session that is the closest to the current one. The score that estimates the interest of a query for the session is computed based on the proximity of the current session with the log queries [41] or based on the probability to have the recommended query following the current query [42, 43].

Hybrid Methods

The only work in this category is [44, 45]. In this work, that assumes a fix data warehouse instance, the query log is processed to detect discovery driven analysis sessions. Sessions are associated with a goal, and recommendations are those queries of former sessions having the same goal as that of the current session. More precisely, the model of expectation is the one proposed in [33, 39, 4]. The log is processed to discover pairs of facts that show a significant difference (like e.g., a drop of sales from one year to the following year). Those pairs are then arranged into a specialisation relation based on the hierarchies. At query time, if the current query investigates a pair that relates to the pairs discovered in the log, then the past queries featuring such pairs are recommended. The main difference with the techniques of [33, 39, 4] is that only the log is searched for interesting deviations.

Example 15. Suppose that the current query result shows that there is a big drop of sales in the sale of Cheese in Europe from year 2009 to year 2010. Suppose that in the log, it is found that past queries have already investigated a big drop of Cheese sales in France for the same couple of years. Then such queries will be recommended.

3.6 Conclusion

This lecture introduced OLAP query personalisation and recommendation, by first defining the basic concepts and then providing an overview of the existing approaches in relational and multidimensional databases. We restricted the scope of this presentation to the problem of, given a database query q,

compute a query $q' \subseteq q$ (personalisation) or compute a query q' such that neither $q' \subseteq q$ nor $q \subseteq q'$ (recommendation). We also categorise the approaches in terms of formulation effort, prescriptiveness, proactiveness, expressiveness, and the type of data needed: current state, history based, external data. With respect to this categorisation, it can be noted that many interesting combinations could investigated. For instance, to the best of our knowledge, there exists no approach that requires a low formulation effort, being proactive and not prescriptive while remaining highly expressive.

One of the main limitations in this field of research is that assessing effectiveness of the approach is very difficult since it should be based not only on real data but also on users' feedback, both being very difficult to obtain due to contexts the typically involves sensitive data. This is for instance illustrated by the fact that, in a data warehouse context, there exists no categorisation of the users' navigational behaviours, whereas such a categorisation exists in the web [46].

The domain of query personalisation and recommendation is still in its infancy. A lot of open issues can be listed, ranging from user privacy to the quality of recommendations and personalisations, most of these issues being also relevant beyond the data warehouse context. For instance, the learning and revision of preferences or navigational habits is one of these challenges. It is of paramount importance for computing more accurate and reliable personalisations and recommendations. A preliminary work tailored to the data warehouse context [47] is a first step in that direction.

References

1. The Economist: A special report on managing information: Data, data everywhere (February 2010), http://www.economist.com/node/15557443
2. Jagadish, H.V., Chapman, A., Elkiss, A., Jayapandian, M., Li, Y., Nandi, A., Yu, C.: Making database systems usable. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, pp. 13–24 (2007)
3. Chen, P.Y., Wu, S.Y., Yoon, J.: The Impact of Online Recommendations and Consumer Feedback on Sales. In: Proceedings of the International Conference on Information Systems, ICIS 2004, Washington, DC, USA, December 12-15, pp. 711–724 (2004)
4. Sarawagi, S.: User-Adaptive Exploration of Multidimensional Data. In: VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt, September 10-14, pp. 307–316 (2000)
5. Rizzi, S.: New Frontiers in Business Intelligence: Distribution and Personalization. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 23–30. Springer, Heidelberg (2010)
6. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
7. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill (2009)

8. Marcel, P., Negre, E.: A survey of query recommendation techniques for data warehouse exploration. In: 7èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2011), Clermont-Ferrand, Paris, Hermann. RNTI, vol. B-7, pp. 119–134 (June 2011)
9. Marcel, P.: Personalization and recommendation of OLAP queries (2011), http://www.info.univ-tours.fr/~marcel/BD/ebiss2011.pptx
10. Stefanidis, K., Koutrika, G., Pitoura, E.: A survey on representation, composition and application of preferences in database systems. ACM Trans. Database Syst. 36(3), 19 (2011)
11. Kießling, W.: Foundations of Preferences in Database Systems. In: VLDB 2002: Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, Hong Kong, China, pp. 311–322 (2002)
12. Chomicki, J.: Preference formulas in relational queries. ACM Trans. Database Syst. 28(4), 427–466 (2003)
13. Adomavicius, G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. Knowl. Data Eng. 17(6), 734–749 (2005)
14. Goel, S., Broder, A.Z., Gabrilovich, E., Pang, B.: Anatomy of the long tail: ordinary people with extraordinary tastes. In: Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, pp. 201–210 (2010)
15. Adomavicius, G., Tuzhilin, A., Zheng, R.: REQUEST: A Query Language for Customizing Recommendations. Information Systems Research 22(1), 99–117 (2011)
16. Kadlag, A., Wanjari, A.V., Freire, J., Haritsa, J.R.: Supporting Exploratory Queries in Databases. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 594–605. Springer, Heidelberg (2004)
17. Mishra, C., Koudas, N.: Interactive query refinement. In: EDBT 2009: Proceedings of 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, pp. 862–873 (2009)
18. Golfarelli, M.: Personalization of OLAP queries. In: 6émes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2010), Djerba, Tunisie, Toulouse, Cépadués. RNTI, vol. B-6 (June 2010); 1 Article invité
19. Stefanidis, K., Drosou, M., Pitoura, E.: "You May Also Like" Results in Relational Databases. In: Amer-Yahia, S., Koutrika, G. (eds.) PersDB 2009, 3rd International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases in Conjunction with VLDB 2009, Lyon, France, August 28, pp. 37–42 (2009)
20. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, April 2-6, pp. 421–430 (2001)
21. Koutrika, G., Ioannidis, Y.E.: Personalization of Queries in Database Systems. In: Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, Boston, MA, USA, March 30- April 2, pp. 597–608 (2004)
22. Koutrika, G., Ioannidis, Y.E.: Personalized Queries under a Generalized Preference Model. In: Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, Tokyo, Japan, April 5-8, pp. 841–852 (2005)
23. Chatzopoulou, G., Eirinaki, M., Polyzotis, N.: Query Recommendations for Interactive Database Exploration. In: Winslett, M. (ed.) SSDBM 2009. LNCS, vol. 5566, pp. 3–18. Springer, Heidelberg (2009)

24. Akbarnejad, J., Eirinaki, M., Koshy, S., On, D., Polyzotis, N.: SQL QueRIE Recommendations: a query fragment-based approach. In: Catarci, T., Stavrakas, Y. (eds.) PersDB 2010, 4th International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases in Conjunction with VLDB 2010, Singapore, September 13 (2010)
25. Khoussainova, N., Kwon, Y., Balazinska, M., Suciu, D.: SnipSuggest: Context-Aware Autocompletion for SQL. PVLDB 4(1), 22–33 (2010)
26. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record 26(1), 65–74 (1997)
27. Sarawagi, S., Agrawal, R., Megiddo, N.: Discovery-Driven Exploration of OLAP Data Cubes. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 168–182. Springer, Heidelberg (1998)
28. Rizzi, S.: OLAP preferences: a research agenda. In: DOLAP 2007: Proceedings of ACM 10th International Workshop on Data Warehousing and OLAP, Lisbon, Portugal, November 9, pp. 99–100 (2007)
29. Bellatreche, L., Giacometti, A., Marcel, P., Mouloudi, H., Laurent, D.: A personalization framework for OLAP queries. In: DOLAP 2005: Proceedings of ACM 8th International Workshop on Data Warehousing and OLAP, Bremen, Germany, November 4-5, pp. 9–18 (2005)
30. Golfarelli, M., Rizzi, S.: Expressing OLAP Preferences. In: Winslett, M. (ed.) SSDBM 2009. LNCS, vol. 5566, pp. 83–91. Springer, Heidelberg (2009)
31. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000)
32. Microsoft: MDX reference (2009), http://msdn.microsoft.com/
33. Sarawagi, S.: Explaining Differences in Multidimensional Aggregates. In: VLDB 1999: Proceedings of 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, UK, September 7-10, pp. 42–53 (1999)
34. Ordonez, C., Chen, Z.: Evaluating Statistical Tests on OLAP Cubes to Compare Degree of Disease. IEEE Transactions on Information Technology in Biomedicine 13(5), 756–765 (2009)
35. Biondi, P., Golfarelli, M., Rizzi, S.: myOLAP: An Approach to Express and Evaluate OLAP Preferences. IEEE Trans. Know. Data Eng. 23(7), 1050–1064 (2011)
36. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: Preference-Based Recommendations for OLAP Analysis. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 467–478. Springer, Heidelberg (2009)
37. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: A Framework for OLAP Content Personalization. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 262–277. Springer, Heidelberg (2010)
38. Cariou, V., Cubillé, J., Derquenne, C., Goutier, S., Guisnel, F., Klajnmic, H.: Built-In Indicators to Discover Interesting Drill Paths in a Cube. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2008. LNCS, vol. 5182, pp. 33–44. Springer, Heidelberg (2008)
39. Sathe, G., Sarawagi, S.: Intelligent Rollups in Multidimensional OLAP Data. In: VLDB 2001: Proceedings of 27th International Conference on Very Large Data Bases, Roma, Italy, September 11-14, pp. 531–540 (2001)
40. Giacometti, A., Marcel, P., Negre, E.: A framework for recommending OLAP queries. In: DOLAP 2008: Proceedings of ACM 11th International Workshop on Data Warehousing and OLAP, Napa Valley, California, USA, October 30, pp. 73–80 (2008)

41. Giacometti, A., Marcel, P., Negre, E.: Recommending Multidimensional Queries. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 453–466. Springer, Heidelberg (2009)
42. Sapia, C.: On Modeling and Predicting Query Behavior in OLAP Systems. In: Proceedings of the Intl. Workshop on Design and Management of Data Warehouses, DMDW 1999, Heidelberg, Germany, June 14-15, p. 2 (1999)
43. Sapia, C.: ROMISE: Predicting Query Behavior to Enable Predictive Caching Strategies for OLAP Systems. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 224–233. Springer, Heidelberg (2000)
44. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query recommendations for OLAP discovery driven analysis. In: DOLAP 2009: Proceedings of ACM 12th International Workshop on Data Warehousing and OLAP, Hong Kong, China, November 6, pp. 81–88 (2009)
45. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query Recommendations for OLAP Discovery-Driven Analysis. IJDWM 7(2), 1–25 (2011)
46. Broder, A.Z.: A taxonomy of web search. SIGIR Forum 36(2), 3–10 (2002)
47. Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., Turricchia, E.: Mining Preferences from OLAP Query Logs for Proactive Personalization. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 84–97. Springer, Heidelberg (2011)

4

The GoOLAP Fact Retrieval Framework

Alexander Löser, Sebastian Arnold, and Tillmann Fiehn

FG DIMA
Technische Universität Berlin
Berlin, Germany
firstname.lastname@tu-berlin.de

Summary. We discuss the novel problem of supporting analytical business intelligence queries over web-based textual content, e.g., BI-style reports based on 100.000s of documents from an ad-hoc web search result. Neither conventional search engines nor conventional Business Intelligence and ETL tools address this problem, which lies at the intersection of their capabilities. Three recent developments have the potential to become key components of such an ad-hoc analysis platform: significant improvements in cloud computing query languages, advances in self-supervised keyword generation techniques and powerful fact extraction frameworks. We will give an informative and practical look at the underlying research challenges in supporting "Web-Scale Business Analytics" applications that we met when building GoOLAP, a system that already enjoys a broad user base and over 6 million objects and facts.

4.1 Introduction

Which companies collaborate with Boeing? Are these organizations also collaborating with Airbus or Fokker? Do employees of these companies have a criminal record? Who published the information?

Each day new pages emerge in the Web that may contain a textual representation of facts for answering these questions. Strategic decision makers may frequently research the Web for questions like these. Often answers to these queries might not be published in textual or structured form by Web 'information aggregators' like *Wikipedia.com*, *Freebase.com*, *Trueknowledge.com* or *Yago* [1]. Rather, this rare factual information is hidden in unstructured Web text on a few Web pages of news agencies or blogs. Unfortunately, collecting factual answers from these pages with a general Web search engine is still a dreaded process for a user.

One option to populate a fact base is to crawl a large document collection. For instance *Google Squared* [2] populates its data base with facts from the large corpus of the general Web. The system extracts these facts from tables, lists and from text with open information techniques. However, in [3] we observed that only a small fraction of a large archive *de facto* contains factual

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 84–97, 2012.

information. Hence, strategies that might execute a full scan over the entire archive can drastically waste processing and storage resources (see also [4]). Another option is discovering facts in retrieved pages from ad-hoc keyword search. Unfortunately, this is still a tedious task, since Web search engines do not return aggregated factual information. The heuristic of a search engine user is: type in keywords as queries, 'extract' relevant facts from the top-k documents, filter out relevant facts and compile facts into a structured fact table. Therefore the user typically repeats this process multiple times in order to *complement missing attribute values* and to enhance the chance to *discover unseen facts*.

Significance of our approach. We present GoOLAP, a system that aims to *automate the fact retrieval process from Web search engines*. GoOLAP has three significant aspects: (1) It provides *powerful operators for analytical Web research on textual information*, such as augmenting facts for an object, tracing back the textual origin of a fact or comparing factual information for a list of objects. (2) As a natural alternative to crawling a large proportion of the Web, GoOLAP *interprets user interactions as input to identify missing facts*. These user interactions *trigger a fact retrieval process* with the goal to populate the GoOLAP fact base from Web-scale indices of existing search engines in an ad-hoc fashion. This process is powered by the *FactCrawl* engine that leverages sophisticated *keyword generation techniques* [3, 5] and *page classification techniques* [6] to retrieve only pages that likely contain missing and rare factual information [7]. (3) GoOLAP combines these approaches to drastically avoid crawling, indexing and extracting potentially billions of irrelevant pages. GoOLAP's human-machine machine generated fact base nearly reaches 6 million facts and objects; a dimension that is still an order of magnitude smaller than community-generated fact bases, such as *DBpedia* [8] or *CrunchBase* [9]. However, our prototype extracts facts in a hybrid approach: It leverages the crowd, the power of GoOLAP users, to discover new objects in queries, extracts and augments with the help of parallel running machines new factual information and again utilizes the power of GoOLAP users to verify and re-trigger this process, for instance in the case of incomplete information.

4.2 Related Work

We discuss relevant related work in the areas of focused fact retrieval from full text indexes, Open Information Extraction and fact search engines.

Keyword query generation with QXtract. The authors of QXtract [10] pioneered work on automatically generating keyword phrases for fact retrieval from full-text indices. The system executes the following stages in a one time learning process: sample seed documents, observe/score phrases in these

documents, query a search engine with the most promising phrases and forward the retrieved pages to an information extractor.

Sample seed documents. QXtract requires a small set of manually specified 'seed' facts to retrieve an initial set of sample pages for training. The size of this sample influences the precision of the learning process; the authors suggest between 100 and 2500 pages. After sampling, the pages are forwarded to the extractor to identify which documents are relevant for the extraction task.

Observe and score phrases. QXtract utilizes three classification approaches for observing and scoring unigram terms from the documents in the seed sample: An SVM-based classifier, the OKAPI system [11] and the rule-based classifier Ripper [12] are trained on the set of seed documents. Each approach returns a list of terms, ordered by their significance for classifying relevant and irrelevant documents. The authors of QXtract propose a threshold based technique to assemble and select relevant phrases as keyword queries out of these terms.

Determine document retrieval order. Next, QXtract retrieves a ranked set of documents from the index for each generated keyword query, where the queries are selected from the three lists in a round robin fashion. All documents retrieved for a phrase are forwarded to the fact extractor. The process continues with selecting the next highest ranked phrases from each list and forwarding the resulting documents to the extraction service. It terminates once no more phrases exist in any list. The authors evaluated their approach on the two fact extractors *CompanyHeadquarter* and *DiseaseOutbreak* and achieved a recall of about 20% while processing 10% of the documents in the test corpus. In [4] the authors describe a framework incorporating the idea of automatic keyword generation as an execution strategy.

Phrase generation from SQL queries. Authors of [13] extract and rank potential keywords from attribute names and values of a structured query. They apply a greedy approach that selects terms based on their relevance measures informativeness and representativeness. Our approach makes use of a similar idea: We trust that Web page authors develop a set of common words and grammatical structures to express important factual information in natural language.

Self-supervised keyword generation. In [5] we published an approach that applies a self-supervised keyword generation method to extract meaningful phrases which often correlate with factual information in sentences. Our approach utilizes open information extraction techniques to generalize fact-type-independent lexico-syntactic patterns from sentences [14]. Recently, we presented an extended version of this work on fact retrieval [3], called

FactCrawl. Our framework allows for the integration and evaluation of different feature generation methods.

Fact extraction. Multiple projects, such as CIMPLE [15], AVATAR [16], DIAL [17], DoCQS [18], PurpleSox [19] describe an information extraction plan with an abstract, predicate-based rule language. Our approach tries to 're-engineer' common patterns from observing fact extractors and other patterns that appear frequently with factual information on a page as features.

Fact search. Recently, the commercial search engine *Google Squared* [2] populated a fact base with machine extracted information from Web tables and Web lists. That prototype has access to any page in the Google index that contains a table or a list. Contrary, GoOLAP can neither rely on nor compete with this massive amount of data. Rather, we utilize user triggered interactions to generate keyword queries only for interesting and frequently requested facts. These keyword queries leverage the index of a Web search engine and retrieve only pages that likely contain facts from which we aggressively filter our irrelevant pages. With both techniques, generated keywords and aggressive Web text filters, we leverage the crawling power of a Web search engine, but can drastically limit our fact extraction costs and avoid crawling and indexing potentially billions of irrelevant pages. Google Squared focuses on facts that are either represented as Web lists or as Web tables [2]. Thereby Google Squared may miss important facts in the much more common textual representation. Our extraction approach is complementary since we focus on natural language text only.

Open information extraction and fact search. The Open Information Extraction approach [14] of Google Squared moves the semantic interpretation of fact types to a human user. Contrary, GoOLAP relies on a predefined set of relation extractors. These extractors return well defined semantics for domain specific facts. Thus, this approach allows the aggregation and interpretation of extracted facts by a machine, such as by an OLAP query processor. In addition, these facts can be integrated with user contributed fact bases, such as the project DBpedia [8].

Combining OLAP cubes and Web Documents. The idea of combining OLAP cubes and web documents is not new. In fact, many digital libraries like ACM, DBLP and Microsoft Research compute aggregations on document meta data. Authors of [20], [21] and [22] published principles for integrating OLAP Cubes and Web documents. Our work utilizes these approaches and goes one step beyond: Our design allows users to trigger a focused fact retrieval process that allows the user to create domain specific specific OLAP cubes, to identify missing facts and to restart this process until the user believes that sufficient facts exist. Thereby the system executes and controls the the retrieval of missing facts while the user exploits OLAP cubes.

The relation to Linked Open Data. Much factual information is readily available as LOD (in RDF) [8] which can probably significantly increase the quality of the results of the GoOLAP system described. Alternatively, GoOLAP can also contribute newly extracted facts to the LOD cloud.

4.3 Exploring GoOLAP's Fact Base

This section describes how a user may explore the GoOLAP fact base; see also [23, 24, 3] for a discussion of the research challenges and [25] for an overview of exploratory search. In this process the system retrieves *documents*, such as HTML pages, issues *keyword queries* against a Web search engine, such as sequences of lexical tokens, and stores, documents, facts, references to documents and the retrieval date in a RDBMS that follows the database layout of [16] (see also next Section).

In the rest of this section we introduce five operations that enable a user to interact with a system. These operators reflect the typical process of an exploratory search *interpret* the query, retrieve *augmented* facts for the query, *expand* a single fact into a list of related facts, and *trace back* the origin if a single fact:

Interpret. A query typed into a search field can express different intentions. For example, a user has the intention *Augment facts for Boeing*, so he enters "Boeing" into the search field. While he is typing the query, the system returns multiple interpretations in an auto complete dropdown, such as *company Boeing*, *person Boeing* or *product Boeing 747*. The user may select an interpretation for *Boeing* (e.g. *company Boeing*) to send the interpreted query to the system.

Augment. This operation collects and augments information for a particular object. In addition, the system returns images of the selected object and shows links to objects of the same type, such as other companies. Finally, the system returns a list of ranked facts. This process has two steps. (1) Fact type enumeration. The system takes as input the selected object. Each object corresponds to a specific type for which the system looks up known fact types in a system-wide ontology. Currently this ontology holds over 70 different fact types. For instance, for an object of the type *company*, the system returns fact types such as *layoffs*, *competitors* or *products*. (2) Return most interesting facts for each type. For each fact type the system queries the corresponding table for non-empty facts. Next, it ranks facts for each type by the notion of interestingness [26]. For the exact process of object reconciliation we rely on existing approaches, such as [27] and [28]. Note, that this functionality is still major challenge when integrating data coming from different (probably inconsistent) data sources.

Expand. This operation enables a comparative analysis across facts for multiple objects. Consider a user who wants to compare different aircraft vendors, such as *Boeing*, *Fokker* or *Northrup Grumman*. For each object in this user defined selection our system projects a set of facts per type and creates a tabular fact sheet. Objects are displayed as rows, fact types as columns. Each cell lists facts ranked by interestingness. Note that this table represents a non-first-normal-form of the underlying fact base; therefore multiple values per cell may exist.

Trace back. For each fact, the user can access the original document at any point of the assessment process in order to continue his research.

Subscribe. Each user has the possibility to register a profile on the GoOLAP site. This enables a registered user to give feedback to the results by clicking on *agree* or *disagree* buttons or editing factual data in-place. If a user is interested in a particular object, e.g. his own company, he can subscribe to the object by clicking on the star next to the object's name. The system then tries to fetch more data for the subscribed object.

4.4 User-Triggered Fact Retrieval from an Inverted Index

The main goal of our framework is to be able to mine all the facts contained in a document collection while processing as few documents as possible. To solve this task we adapt keyword based document retrieval systems, such as Web search engines, to retrieve only those pages that contain a fact (a factual statement expressed in unstructured text which an information extractor can detect and extract). Our fact retrieval method emulates a search engine users behavior to solve the "inverse retrieval" problem of identifying discriminative keywords for the retrieval of relevant documents. Figure 4.1 introduces our general fact retrieval process:

Step 1: Interpret and execute user query. The system collects user interactions, determines the intent of the user and executes a query against the local fact base. While a crawl-based fact retrieval execution guarantees that all documents in an archive are processed, an indexed-based execution might miss some relevant documents. For compensating this disadvantage, GoOLAP observes the *completeness* and *rarity* [7] of returned results to identify potentially missing facts. If necessary, GoOLAP schedules a fact retrieval activity (see Table 4.1). For example, if a user requests facts for an object that is not yet contained in the GoOLAP base, the system will trigger the retrieval of pages that contain facts for the missing object.

Step 2: Generate keyword query to retrieve missing facts. GoOLAP forwards the corresponding textual object representation and the semantic

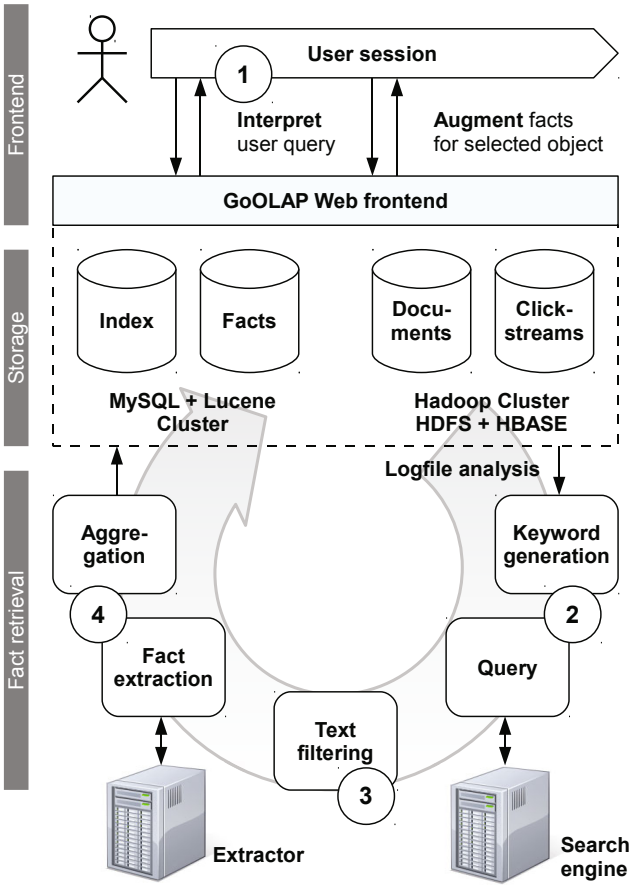


Fig. 4.1. User triggered fact retrieval from Web search engines. The fact retrieval process is interactively triggered by user interaction, here INTERPRET and AUGMENT.

type of the missing fact to FactCrawl [3], a framework that emulates a search engine user's behavior to solve an inverse retrieval problem. This framework queries a Web search engine with automatically generated keywords, re-ranks the resulting list of URLs according to a novel fact score and forwards only promising documents to a fact extractor. FactCrawl generates keywords using structural, syntactic, lexical and semantic information from sample documents. Thereby FactCrawl estimates the fact score of a document by combining the observations of keywords in the document.

Step 3: Filter out irrelevant pages. GoOLAP utilizes a fact predictor [6]; this technique reliably filters out irrelevant pages and only forwards relevant

Table 4.1. GoOLAP collects and interprets more than 30 user interactions that may trigger a fact retrieval process. This table displays user interactions and system activities contained in our demonstration scenario.

User interaction	**Intention:** The user...	**System activity:** The system...	**Retrieval activity:** FactCrawl...
Click a link that points to GoOLAP	discovers a GoOLAP result page for *Company:Boeing* in another search engine and clicks on the link	displays the AUGMENT result page for *Company:Boeing*	retrieves more facts for *Company:Boeing*, such as *headquarter*, *layoffs* etc.
Search with keyword	wants to augment facts for the object *Company:Boeing* and types *Boeing* into the search field	interprets the input and displays an autocomplete dropdown with multiple interpretations, such as *Company:Boeing*, *Product:Boeing 747* etc.	ranks the order of interpretations when the user selects *Company:Boeing*
Search missing object	wants to augment facts for an object *X* that is not yet contained in the GoOLAP fact base	displays a message to inform the user about the empty result and proposes a subscription of *X*	retrieves facts for the object *X* with high priority
INTERPRET keyword search via autocomplete	wants to augment facts to the object *Company:Boeing* displayed in the autocomplete and clicks on it	augments facts to *Company:Boeing* and displays the AUGMENT page for that object	ranks the popularity of *Company:Boeing* and retrieves more facts, such as *headquarter*, *layoffs* etc.
Show AUGMENT result	requests an AUGMENT result for the object *Company:Boeing*	displays the top rated facts for *Company:Boeing* and three arbitrary documents containing information about *Boeing*	ranks the popularity of *Company:Boeing* and retrieves more facts
TRACE BACK the document	wants to trace back the textual origin of the fact *EADS competes Boeing* and clicks on the document symbol	displays a snippet from the source document and explains the origin of a fact by highlighting the sentence where the fact was extracted from	ranks the popularity of the fact *EADS competes Boeing* and tries to retrieve more evidences of this fact
EXPAND list of objects	wants to compare *Competitors of Boeing* and clicks on the EXPAND icon in the column header	expands the list of *Competitors of Boeing* and displays an interactive table view for comparison	retrieves more facts for *Company:Boeing* and the *competitive* relation to *Airbus*, *EADS*, *Fokker* etc.

pages to the GoOLAP fact extractors. Most importantly, our fact predictor is two orders of magnitude faster than the fact extractor. The fact predictor is based on a support vector machine that evaluates pages on a sentence level, where each sentence is transformed into a token representation of shallow text features.

Step 4: Fact extraction. GoOLAP extracts factual information and resolves objects with home-grown extractors based on [14] and commercial extractors, such as OpenCalais [29]. Overall, GoOLAP provides more than 70 different fact extractors for the domains people, companies, media and entertainment. As a result, the system wide ontology is limited to the set of fact types provided by the extractor, but also open to previously unseen fact instances that the fact extractor discovered in retrieved documents. An interesting open research challenge is the alignment of our fact base with DBpedia / Freebase or any other large scale factual knowledge base. OpenCalais may provides only marginal quality on texts from domains that are different from business and news, such as for medical documents. Therefore our system is open to additional fact extractors such as *extractiv.com* or *alchemy.com*. The retrieval process then aggregates the extracted facts and stores them into the local fact base. Selecting those extractors which provide best performance on a certain domain, when there are alternatives available, is an open challenge.

Data storage and parallel execution engine. Our execution and storage system bases on a hybrid solution is hybrid, along traditional approaches, such as [20], [16]. Even though it would be more interesting to have a single system, we chose this hybrid approach for the following reasons:

- *Factual information is small compared to the raw text documents.* Therefore we chose to store the local fact base, the ontology, objects and small

document snippets of a few 100 bytes in a horizontally partitioned MySQL cluster database on a raid-based 256 GB solid state disc (SSD) array. The technology allows GoOLAP users to retrieve as well as to update and rate facts.

- The current user base may generate interactions that trigger the retrieval of *several hundred thousand pages* daily. Current RDBMS may only hold the documents for several hundred days with a reasonable cost/storage balance. Therefore, we manage the sheer mass of raw textual documents in a *Hadoop* [30] cluster, using commodity hardware, HDFS and HBASE. The MySQL cluster references these documents via unique keys.
- *Fact extraction is computational processing intensive.* GoOLAP processes the massive amounts of requests for new facts in a parallel fashion on a cluster of 12 nodes, each running with 4 GB RAM on a 2x 2.8 GHz Intel CPU. The parallel query execution engine bases on Hadoop. We chose to abstract the functional programming interface of Hadoop through the declarative query language *JAQL* [31] and extend JAQL with first-order functions for keyword generation with FactCrawl [3], Web text filtering [6] and fact extraction [29]. JAQL represented objects, pages, facts and interactions with the semi-structured JSON format. Jaqls design has been influenced by Pig [32], Hive [33], DryadLINQ [34], among others, but has a unique focus on the following combination of core features: (1) a flexible data model, (2) reusable and modular scripts, (3) the ability to specify scripts at varying levels of abstraction, referred to as physical transparency, and (4) scalability (see also).

4.5 Demonstration Scenario

Our interactive online demo shows the interplay of users exploring the GoOLAP fact base and thereby triggering new requests for retrieving missing facts from the index of a Web search engine. Figure 4.2 shows our demonstration scenario. Our prototype can be reached at www.goolap.info.

Consider an analyst who wants to have an overview over competitors of Boeing. *On the GoOLAP start page, she begins to type "Boeing" into the search field* (1). *The auto complete dropdown returns multiple interpretations of Boeing, picking the* company *object by default. After sending the query, the following page presents an overview over augmented facts of different types (e.g. business location, products or employment) for the company Boeing and some arbitrary documents that include information about that object. She now opens the table* Business: Competitor *that shows the top 15 ranked facts about competing companies* (2). *We suppose she is unsure whether* EADS *is a competitor of Boeing, thus she clicks on the document symbol on the right of the row to prove evidence for that fact. The next page shows a document from www.defenseprocurementnews.com which explains "Boeing and EADS are in direct competition"* (3). *She can help ranking the fact by clicking on*

Fig. 4.2. A typical user session with the goal to collect facts about competitors of *company Boeing*

the agree *button or find the sources of other facts in the table the same way. Next, the analyst desires to compare the competitors. She clicks on the button in the column header of the competing companies to expand the list. The view presents a tabular list of competitors of Boeing* (4)*. She can manually remove an incorrect object in the list or add another one that is missing. Suppose the analyst wishes to compare the employee size of the companies. She performs an outer join on the table by adding the column* Business: Employee Size *from the dropdown menu in order to get a comparable result of facts* (5)*. If there is a wrong result in one cell, she can display alternative values and find a better one. Again, she can trace back the evidence of a fact by clicking the document symbol. She is now happy with her result and saves the table under the name "Aviation Companies". Now, she can continue her research at a later time or share the table to other users, allowing them to make their own changes.*

4.6 Research Challenges

GoOLAP presents our attempt in providing a versatile approach for interactive fact retrieval from the Web. We believe that studying the new types of Web usage and the interplay of fact exploration with fact retrieval – a well-known important problem in information retrieval – will open up a lot of new challenges and opportunities:

Ranking fact sources. Only very few domains provide most of the facts in GoOLAP (see also Figure 4.3). Often these *fact aggregators* aim to collect factual information from news articles or human collaborations. Worse, in [3] we discovered that most fact types appear only in less than 1% of retrieved documents. An interesting direction is a systematic integration of these statistical observations into the fact retrieval process; for instance, we could observe domains about fact update cycles or could estimate common replication strategies across domains.

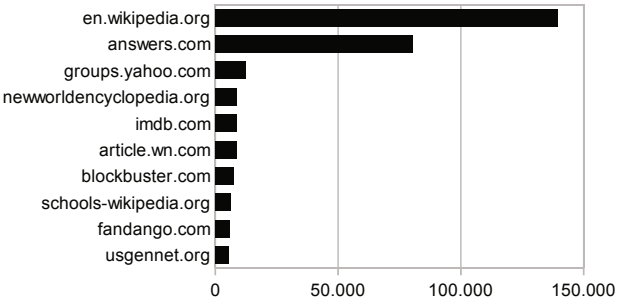


Fig. 4.3. GoOLAP receives facts from 40.414 different Web domains. This distribution shows the top-10 domains that provided most facts. The X-axis depicts the number of extracted facts per domain.

Quality-based fact ranking. GoOLAP ranks different facts for the same object with the measure of interestingness [26]. This measure captures how often authors of retrieved pages mention the same fact for the same object. The design of more meaningful, potentially quality based, ranking metrics desires investigation, such as metrics for ranking 'rarely appearing' facts in documents which are often requested through user interactions.

'Crowd'-based fact sharing and verification. Similar to a data market, users may request the system to collect 'private' fact collections and share these collections with few 'trusted' users. How can the system leverage these user interactions and fact collections for discarding incorrect facts? Can we derive a fact quality measure from observing the sharing of fact collections?

Active learning of labeled examples from user interactions. Developing high-quality information extraction (IE) rules, or fact extractors, is an iterative and primarily manual process, extremely time consuming, and error prone. In each iteration, the outputs of the ex- tractor are examined, and the erroneous ones are used to drive the refinement of the extractor in the next iteration. Recently, authors of [35] introduced a provenance-based solution for suggesting a ranked list of refinements to an extractor aimed at increasing its precision. The ability to automatically generate useful refinements depends on the number, variety and accuracy of the labeled examples provided to the system. In most cases the labeled data must be provided by the rule developer. Unfortunately, labeling data is itself a tedious, time-consuming and error prone process. It would be interesting to investigate whether active learning techniques [36] can be combined with techniques for adapting open information extraction to domain specific relations [37] and GoOLAP click streams to present to the developer only those most informative examples, therefore facilitating the labeling process.

4.7 Conclusion

We introduced GoOLAP.info, a novel class of applications for answering intelligence queries over natural language Web text. GoOLAP provides powerful operators for retrieving, extracting and analyzing factual information from textual representations on the Web. At the core of this service lies FactCrawl, a powerful engine for the retrieval of unseen or missing facts from a Web search engine to a structured fact base. Over time, GoOLAP may evolve to a comprehensive, effective and valuable information source. Unlike systems that utilize a cost intensive crawling strategy (such as Google Squared), GoOLAP elegantly utilizes user interactions, generated keyword queries and text filtering techniques to populate an ad-hoc and incrementally improving fact base. Compared to existing approaches that crawl and maintain very large Web archives, GoOLAP tries to minimize retrieval costs through user triggered fact retrieval.

We outlined exciting challenges, such as the design of an iterative fact retrieval process, the interpretation of user interaction and automatic keyword query generation. As mentioned already, there is a lot of experimental work done and to do in the future to rank or verify information; see also work about ranking interesting assertions in [26], ranking facts [38] and work on identifying the 'true' origin of an information [39].

Finally, GoOLAP gives us access to click streams that allow the academic community to study user interactions and to design novel powerful Web research operators. These challenges should appeal to and benefit from several research communities, most notably, the database, text analytics and distributed system worlds.

References

1. Kasneci, G., Suchanek, F.M., Ramanath, M., Weikum, G.: The YAGO-NAGA Approach to Knowledge Discovery. SIGMOD Record 37(4) (2008)
2. Crow, D.: Google Squared: Web scale, open domain information extraction and presentation. In: Proceedings of the 32nd European Conference on IR Research, ECIR 2010 (2010)
3. Boden, C., Löser, A., Nagel, C., Pieper, S.: Factcrawl: A fact retrieval framework for full-text indices. In: Proceedings of the 14th International Workshop on the Web and Databases, WebDB (2011)
4. Ipeirotis, P.G., Agichtein, E., Jain, P., Gravano, L.: To search or to crawl?: towards a query optimizer for text-centric tasks. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD 2006, pp. 265–276. ACM, New York (2006)
5. Löser, A., Nagel, C., Pieper, S.: Augmenting Tables by Self-Supervised Web Search. In: 4th BIRTE Workshop in Conjunction with VLDB (2010)
6. Boden, C., Häfele, T., Löser, A.: Classification Algorithms for Relation Prediction. In: Proceedings of the ICDE Workshops (2010)
7. Löser, A., Nagel, C., Pieper, S., Boden, C.: Self-Supervised Web Search for Any-k Complete Tuples. In: Proceedings of the EDBT Workshops (2010)

8. DBpedia data set, http://wiki.dbpedia.org/Datasets#h18-3 (last visited June 14, 2011)
9. CrunchBase, http://www.crunchbase.com (last visited June 14, 2011)
10. Agichtein, E., Gravano, L.: Querying text databases for efficient information extraction. In: Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE), pp. 113–124 (2003)
11. Robertson, S.E.: On term selection for query expansion. J. Doc. 46, 359–364 (1991)
12. Cohen, W.W.: Fast Effective Rule Induction. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 115–123 (1995)
13. Liu, J.: Answering structured queries on unstructured data. In: Proceedings of the Ninth International Workshop on the Web and Databases, WebDB 2006, pp. 25–30 (2006)
14. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open information extraction from the web. Commun. ACM 51, 68–74 (2008)
15. Shen, W., DeRose, P., McCann, R., Doan, A., Ramakrishnan, R.: Toward best-effort information extraction. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, pp. 1031–1042. ACM, New York (2008)
16. Chiticariu, L., Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F.R., Vaithyanathan, S.: Systemt: an algebraic approach to declarative information extraction. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010, pp. 128–137. Association for Computational Linguistics, Stroudsburg (2010)
17. Feldman, R., Regev, Y., Gorodetsky, M.: A modular information extraction system. Intell. Data Anal. 12, 51–71 (2008)
18. Zhou, M., Cheng, T., Chang, K.C.C.: Docqs: a prototype system for supporting data-oriented content query. In: Proceedings of the 2010 International Conference on Management of Data, SIGMOD 2010, pp. 1211–1214. ACM, New York (2010)
19. Bohannon, P., Merugu, S., Yu, C., Agarwal, V., DeRose, P., Iyer, A., Jain, A., Kakade, V., Muralidharan, M., Ramakrishnan, R., Shen, W.: Purple sox extraction management system. SIGMOD Rec. 37, 21–27 (2009)
20. Chen, Z., Garcia-Alvarado, C., Ordonez, C.: Enhancing document exploration with OLAP. In: Proceedings of the ICDM Workshops, pp. 1407–1410 (2010)
21. Pérez, J.M., Llavori, R.B., Cabo, M.J.A., Pedersen, T.B.: R-cubes: OLAP cubes contextualized with documents. In: ICDE, pp. 1477–1478 (2007)
22. Sismanis, Y., Reinwald, B., Pirahesh, H.: Document-Centric OLAP in the Schema-Chaos World. In: Bussler, C.J., Castellanos, M., Dayal, U., Navathe, S. (eds.) BIRTE 2006. LNCS, vol. 4365, pp. 77–91. Springer, Heidelberg (2007)
23. Löser, A., Hüske, F., Markl, V.: Situational Business Intelligence. In: 3rd BIRTE Workshop in Conjunction with VLDB (2009)
24. Löser, A.: Beyond search: Web-scale business analytics. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE 2009. LNCS, vol. 5802, p. 5. Springer, Heidelberg (2009)
25. Marchionini, G.: Exploratory search: from finding to understanding. Communications of the ACM 49, 41–46 (2006)
26. Lin, T., Etzioni, O., Fogarty, J.: Identifying interesting assertions from the web. In: Proceedings of the 18th CIKM Conference (2009)

27. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, SIGMOD 2005, pp. 85–96. ACM, New York (2005)
28. McCarthy, J.F., Lehnert, W.G.: Using decision trees for conference resolution. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 2, pp. 1050–1055. Morgan Kaufmann Publishers Inc., San Francisco (1995)
29. OpenCalais, http://www.opencalais.com (last visited June 14, 2011)
30. Apache Hadoop, http://hadoop.apache.org (last visited June 14, 2011)
31. Beyer, K.S., Ercegovac, V., Gemulla, R., Balmin, A., Eltabakhy, M., Kanne, C.C., Ozcan, F., Shekita, E.J.: Jaql: A scripting language for large scale semistructured data analysis. Proceedings of the VLDB Endowment 4(12) (2011)
32. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, pp. 1099–1110. ACM, New York (2008)
33. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive- a warehousing solution over a map-reduce framework. In: VLDB 2009: Proceedings of the VLDB Endowment, vol. 2, pp. 1626–1629 (2009)
34. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P.K., Currey, J.: Dryadlinq: a system for general-purpose distributed data-parallel computing using a high-level language. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, OSDI 2008, pp. 1–14. USENIX Association, Berkeley (2008)
35. Liu, B., Chiticariu, L., Chu, V., Jagadish, H.V., Reiss, F.: Automatic rule refinement for information extraction. PVLDB 3(1), 588–597 (2010)
36. Thompson, C.A., Califf, M.E., Mooney, R.J.: Active learning for natural language parsing and information extraction. In: Proceedings of the Sixteenth International Conference on Machine Learning, ICML 1999, pp. 406–414. Morgan Kaufmann Publishers Inc., San Francisco (1999)
37. Soderland, S., Roof, B., Qin, B., Xu, S., Mausam, E.O.: Adapting open information extraction to domain-specific relations. AI Magazine 31(3), 93–102 (2010)
38. Jain, A., Pantel, P.: Factrank: random walks on a web of facts. In: Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010, pp. 501–509. Association for Computational Linguistics, Stroudsburg (2010)
39. Dong, X.L., Srivastava, D.: Large-scale copy detection. In: Proceedings of the 2011 International Conference on Management of Data, SIGMOD 2011, pp. 1205–1208. ACM, New York (2011)

5

Business Intelligence 2.0: A General Overview

Juan Trujillo and Alejandro Maté

Lucentia Research Group
Department of Software and Computing Systems
University of Alicante
Spain
{jtrujillo,amate}@dlsi.ua.es

Summary. Business Intelligence (BI) solutions allow decision makers to query, understand, and analyze business data in order to make better decisions. However, as the technology and society evolve, faster and better informed decisions are required. Nowadays, it is not enough to use only the information from the own organization and making isolated decisions, but rather requiring also to include information present in the web like opinions or information about competitors, while using collective intelligence, collaborating through social networks, and supporting the BI system with cloud computing. In response to this situation, a vision of a new generation of BI, BI 2.0, based on the evolution of the web and the emerging technologies, arises. However, researchers differ in their vision of this BI evolution. In this paper, we provide an overview of the aspects proposed to be included in BI 2.0. We describe which success factors and technologies have motivated each aspect. Finally, we review how tool developers are including these new features in the next generation of BI solutions.

Keywords: Data warehouses, business intelligence 2.0, web 2.0, real-time, cloud computing, collaborative BI, social networks, collective intelligence, crowdsourcing.

5.1 Introduction

Over the last decade, the use of Business Intelligence (BI) solutions has been steadily increasing. Even in the recent recession period, a study from the Gartner Group showed that the BI market not only did not decrease, but instead it grew 4% [1]. BI solutions allow decision makers to query, understand, and analyze business data, in order to make better decisions and gain a competitive edge. Traditionally, BI applications allow managers and decision makers to acquire useful knowledge about the current performance and problems of the business from the data stored in their organization, by means of a variety of technologies. These technologies range from data warehousing, data mining, and OLAP, to business performance management and periodical business reports. Research in these areas has produced consolidated solutions,

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 98–116, 2012.

techniques, and methodologies, and there is a variety of commercial products available that are based on these results.

More recently, a new trend in BI applications has emerged: BI applications no longer limit their analysis to the data of their own organization. Increasingly, they also source their data from the outside, thus complementing internal company data with value-adding information from the Web (e.g. retail prices of products sold by competitors or opinions posted by customers), in order to provide richer insights into the new dynamics of business and to better support decision-making processes. As a result, BI applications aim to assist modern management practices, where decision-making requires a comprehensive view of the market and the business environment as a whole, thus BI solutions using just internal company data no longer suffices. On the other hand, at the same time as data from the Web is being included into BI applications, BI applications are also evolving towards the web. Recently, internal company information systems are being transformed into BI as a service (e.g. hosted BI platforms for small and medium-sized companies) and software support to manage business outsourcing or crowdsourcing is the target of huge investments and the focus of enormous research efforts by both industry and academia.

Nevertheless, this trend is not limited to just relocate the BI processes onto the web, but rather to transform how BI is performed. This transformation is being influenced by the apparition of different, new technologies in the web 2.0, as well as the recent rising of the social networks [2]. Together, these factors provide a general vision of which are the features the next generation of BI tools should include. However, when analyzing the influence of the different factors on BI processes, there is no clear consensus. While some authors [3, 4, 5] focus on the technical aspects and propose to adapt the most recent techniques to the current BI tools, others describe a complete transformation of the BI processes [6, 7]. In our reseach group, and according to this new evolution of classical BI, we have been organizing two editions of the BEWEB Workshop [8, 9] trying to allocate this evolution. Furthermore, in [2], we also collected different trends related to Business Intelligence and the web.

In this paper, we provide an overview of the different aspects of the so-called "BI 2.0", how the web has influenced BI and how BI is reacting to this influence, including the analysis of technical challenges that must be overcomed, as well as how the current BI tools are integrating the new features. We analyze which are the common aspects envisioned, in order to identify the scope of the 2.0 generation. In order to visualize how these new aspects could be integrated into the traditional architecture of a BI system based on data warehousing, we show a potential architecture in figure 5.1. In this figure, we can see how traditional Extraction, Transformation and Loading (ETL) batch processes are now being substituted by real-time integration processes. These processes should also integrate data coming from the web or, alternatively, this data should be integrated into the analysis at later stages, without being stored in the Data

Warehouse (DW). The DW is now distributed in the cloud, which provides the necessary scalability. Furthermore, the information from the DW is retrieved through BI services, which send the requested information through the network to the users. Finally, these users do not take decisions in an isolated manner, but instead interact and interchange information achieving better decisions in shorter times. This interchange can be done either internally, through the business BI system, or between organizations, for example using Business Intelligence Networks [10].

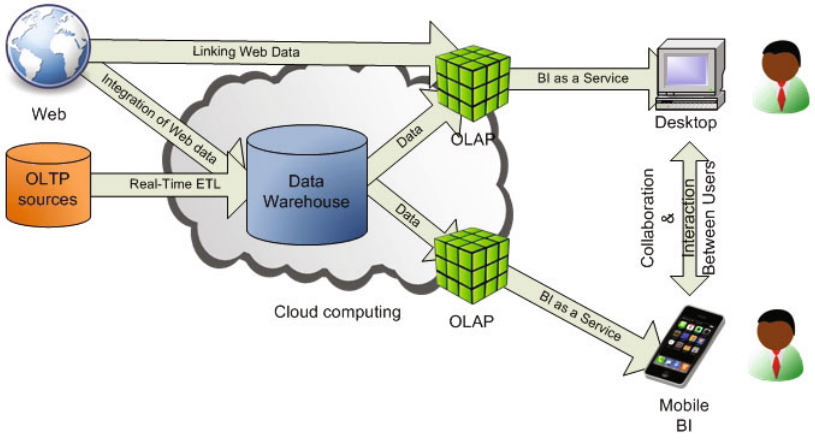


Fig. 5.1. An overview of the new BI 2.0 architecture

The rest of the paper is structured as follows, Section 5.2 gives a brief overview of the recent evolution on BI and DWs. Section 5.3 introduces the basic concepts related to BI 2.0. Section 5.4 describes how the new technologies and the web have influenced traditional BI processes and tools. Section 5.5 describes the current technical challenges and the different solutions proposed to overcome them. Section 5.6 provides an overview of the current BI tools and which new features are being implemented. Finally, Section 5.7 describes the conclusions and sketches the expected evolution in this area.

5.2 Business Intelligence and Data Warehouses in a Nutshell

Traditionally, BI has been involved in analyzing structured data, aiming to improve the business performance and providing a competitive edge. However, due to the size of data to be queried, and the nature of the decisional environment, a specific design for the database which stores the necessary information for the BI process is required [11]. In order to solve this problem, it

is widely accepted that the design of the database schema should be oriented to improve the performance, creating a separate database with a schema that minimizes the number of joins required to answer a given query. The resulting schema at logical level is named Star Schema, and its variants Snowflake Schema and Fact Constellation, depending on how tables are organized. This methodology [12, 11] has become popular for building the DW, and vendors have implemented their own DW designer tools using this approach.

However, a main problem arises from designing the DW from a logical point of view. Since the logical model is designed for storing data and improving query performance, it does not consider the analysts' needs [13]. Therefore, it is hard for an analyst to (i) understand the underlying schema, and (ii) obtain useful information from the stored data. The main drawback is that the logical level lacks multidimensional details, required to specify information related to hierarchies and other complex multidimensional structures, which organize the data for the analysis task and hide implementation details.

In order to solve this problem, researchers have proposed a number of different conceptual modeling approaches [14, 15, 16, 17], although unfortunately, up to now none of them has been accepted as a standard. Nevertheless, vendors have also perceived this defficiency, and include their own representations of multidimensional structures into their tools, in order to allow analysts to query the DW without using SQL queries. However, being able to query the

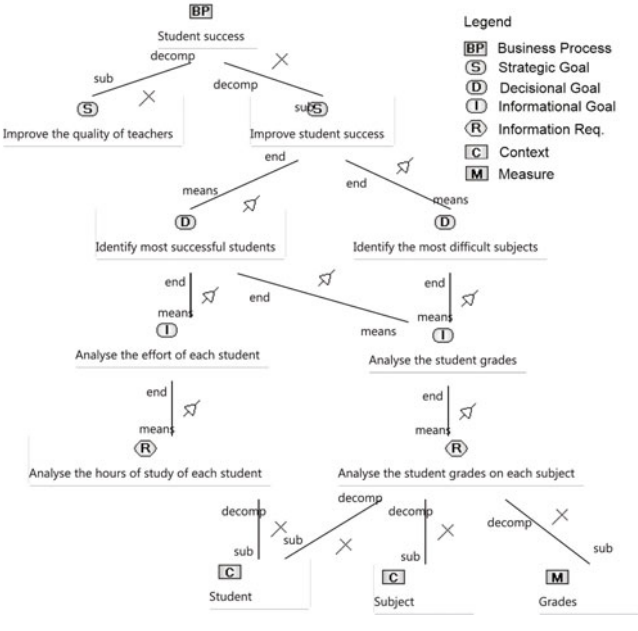


Fig. 5.2. An example of requirements modeling for DWs using the Lucentia BI tool

DW does neither guarantee that it stores the necessary data nor guarantees that it is structured as required by the analysts. Therefore, some researchers have gone a step further, and include a requirements analysis stage in their DW design process [13, 18], providing an alignment between the DW and the analysts' needs, thus reducing the failure rate of DW projects. Most of these requirement analysis stages are based on intentional analysis, where the analysts' needs are modelled as goals. An example of the intentional analysis proposed in [18], using an i* [19] profile, is shown in figure 5.2. This model represents the user intentions starting from a *Business Process*. In order to improve this process, there are a series of goals to fulfill, which are eventually achieved by means of gathering the necessary information (*Contexts* and *Measures*), which will be stored in the DW.

Moreover, in addition to the requirementsnanalysis stage, a hybrid approach for designing DWs by using the Model Driven Architecture, was proposed in [20, 21]. This hybrid approach considers both the user needs, at requirements level, as well as the existing data in data sources, thus detecting any problem or conflict in early stages. Since the proposal was aligned with MDA, it allows the DW designers to build the DW in a semi-automatic way, dramatically cutting development time and costs, and significantly improving the DW development process.

Nevertheless, many challenges are still open for further research in DWs in order to better support the BI processes. On the one hand, BI processes are constantly evolving. Therefore, it is crucial for the DW to include support for new requirements as BI processes change. Some work has been recently done to address this issue [22]. On the other hand, there is still ongoing work on how to include the unstructured information present in the Web into the DW [23, 3, 24, 25]. Information present on the Web can be highly relevant for the business. However, this information is not always accurate and its correctness is not guaranteed. Therefore, integrating it directly into the DW with the rest of information stored by the organization may lower the quality of the data.

Furthermore, the usability of BI tools is still an issue. Some works [26] show that analysts and decision makers do not exploit all the capabilities provided by the BI platforms. This is mainly due to two reasons: (i) these platforms provide a huge set of features, transforming the problem of analyzing huge amounts of data into a problem of finding the right approach (and the right tool) in order to perform the analysis, and (ii) most advanced features provided by BI tools actually require technical knowledge of both the tool and the underlying data structures, e.g. building interactive dashboards, which does not allow analysts to use BI platforms to their full extent. In order to solve this problem, the current direction is providing more user-centric tools, which provide a personalization of the platform while helping the users in their analysis tasks, allowing them to query the data through QA systems and more intuitive interfaces.

After having briefly presented the history about BI & DWs, we will proceed to describe and analyze the upcoming evolution of BI 2.0.

5.3 Basic Concepts Related to BI 2.0

In this section we will introduce the basic concepts related to BI 2.0. Each concept included in this section is either a technology or an aspect related to the new vision of BI.

- **Real-time:** Real-time usually refers to the fact that the maximum time required for a task to be completed is known and defined. However, in BI, real-time often refers instead to the concept of up-to-date data [5], and data streaming. Therefore, when talking about real-time data in BI, we must deal with data flows whose structure is unknown until they are being interpreted. The lack of completeness of the information at a certain point forces to manage exceptional situations which would otherwise not occur. For example, we may have only partial information about a certain transaction, such as the amount, date and client related to a transaction, but not the products.
- **SaaS** (Software as a Service) [4]**:** SaaS refers to software being developed for its use through Service Oriented Architectures (SOA). The Service Oriented Architecture Protocol (SOAP) allows to invoke pieces of software through the HTTP protocol, by passing the parameters specified by the service interface. In this way, SOAP provides a way to invoke software services while abstracting from the implementation technology. Recently, this vision is also being applied to BI solutions. This allows us to obtain results from algorithms, reports and other interesting information from a remote server and combine this services into new value-added ones. The most interesting feature about SaaS is that it allows us to deploy solutions based on cloud computing, which are highly scalable, as opposed to traditional solutions. Additionally, this approach allows companies to "rent" their software and BI services through internet.
- **Cloud computing**[27]**:** Cloud computing was originated as the integration of several, heterogeneus elements into a "cloud" or network. A middleware layer provides a homogeneous interface for the user or software accessing it, while hiding the details of the underlying technology. The cloud also supports the addition of new elements, allowing us to increase the network capabilities to meet the demand as necessary depending on the work load.
- **Collective intelligence**[28]**:** Collective intelligence was first used to refer to emerging behaviours in colonies. Often, these colonies would be formed by insects, and presented more complex behaviours than those of the individuals forming them. An example collective intelligence can be seen especially in social networks, where decentralized groups of people with no leader are able to take decisions and promote initiatives which would not be possible by a single individual.

- **Crowdsourcing[29]:** Crowdsourcing refers to delegating a certain task to the crowd. The effectiveness of this approach comes from the fact that, typically, each individual must only perform little effort in order for the group to achieve its goal. Moreover, the collective intelligence present in the crowd can obtain solutions which could be very difficult to discover by an individual, even if he was dedicated exclusively to that task [6].
- **Social networks:** Social networks consist of a collection of data provided by its own participants, as well as the relationships between them. These networks allow their participants to interact and contribute with further information, thus enriching the existing data. The collaboration between participants provides faster and better results than what a single individual can achieve, thus, researchers are developing proposals which apply this concept to BI. [6, 10].
- **Linked data[30]:** Linked data refers to the idea of relating each piece of information to the rest of information which affects or is affected by that piece. Ideally, linked data means knowing and being able to exploit the existing relationships between every piece of information recorded, which, in turn, means that the relationships are semantically tagged and can be used by a computer to reason. This aspect is specially relevant in order to automatically obtain knowledge from existing information in the web and the information stored in DWs [31].
- **Opinion mining[32]:** Opinion mining refers to the process of describing the general feelings or opinions of a group of people towards a certain element. In this way, opinion mining implies being able to understand a given set of opinions and obtaining a conclusion from them. This information is typically found in the web as unstructured data. Nevertheless, this information can be highly relevant for an organization, enabling to identify which products are perceived better by the customers and why.
- **Process oriented BI[33]:** Process oriented BI is a point of view that focuses on the processes and their logic, relating the stored data to business process performance, instead of focusing on simply presenting aggregated data in different formats. This point of view allows us to identify and restructure processes which are not contributing towards the business goals.

After having introduced the basic concepts, we will proceed to describe how the web, and the new technologies, have influenced the vision of BI and the features included in BI tools.

5.4 Influence from the Web on Business Intelligence

As society evolves, and the ratio of connectivity to the Internet increases, organizations find themselves environment that is constantly and rapidly changing. In the recent years, more and more businesses have started to provide their services in an online fashion. Therefore, now customers have easy access to a wide variety of offers, and become much more critic with the

products they buy. With the appearance of social networks, the blogosphere, and the web 2.0 [34], customers interchange opinions about the products they buy and other similar offers. These opinions influence other customers [32], either motivating them to also purchase that product, or stopping them from buying it. In this context, businesses need to consider as much information as possible when taking strategic decisions [35, 36], in order to gain the edge and retain or even increase their share of the market. Moreover, these decisions must be agile, in order to react in time to existing problems and threats, or covering possible weaknesses. The introduction of these new needs has altered different aspects related to BI tools and processes, such as: user interfaces, up-to-date periods of information, number of persons required to take a decision, how data is presented and the focus of analysis.

The first aspect that has experienced an evolution are the **user interfaces** of the BI solutions. Nowadays, the information required to take decisions must be available to be checked from anywhere. Additionally, the way of interacting with the tools must be intuitive and require as little training as possible. Therefore, desktop applications have become obsolete and web interfaces are almost mandatory. In fact, most BI solutions [37, 38, 39, 40] already include a website as their interface. Recently, these web interfaces have been adapted to be visualized in mobile devices, allowing the appearance of mobile BI solutions, often referred as Mobile BI. Mobile devices are characterized by their reduced size and memory, with interaction mainly based on clicking a touch screen. Therefore, BI 2.0 solutions must provide services which generate user interfaces adapted to show information in a very limited space, while also allowing us to interact and navigate through the data with simple clicks, and transferring only the strictly necessary information.

The second aspect which is changing is the information **up-to-date periods**. Initially, information was typically presented in the form of reports, designed to identify trends and allowing to elaborate strategies in order to avoid future problems. However, nowadays reports are checked when there is already a problem. This is mainly due to two reasons. The first reason is that they are not easy to be read for decision-makers, unless they are being used to identify an existing problem. The reason is that they typically present a huge quantity of non-interactive data, thus they do not allow to perform ad-hoc analysis and its difficult to relate them to business goals and strategies. The second reason is that, very often, they arrive too late due to the rapid change of the environment. In the current environment, monthly and weekly reports are already dated when they are generated. For example, a defective product or a security problem can harm the corporate image in under a week, or even in less than one day. Some examples are the well-known *Dell hell* history or, more recently, the security breaches at Sony. Previously, unattended complaints, and most problems, would just dissapear and other customers would not notice. Now, customers interact with each other and post their opinions, forcing the business to react at the same speed as this information is spread. Therefore, businesses now need to gather and analyze their incoming data in

real-time, and due to its size, in most cases this analysis is required to be done automatically.

The third aspect that is evolving is **how decisions are taken**, inspired by the recently seen social networks. Traditionally, decision-makers would take decisions according to reports provided by the BI system. These decisions would be taken in an isolated manner or in small groups of executives and analysts. Some authors [6, 10] propose that, a better way to take decisions, is to use the collective intelligence of the enterprise or directly delegate them to the crowd (*crowdsourcing*). In this way, when taking a decision, an agile forum of discussion could be created, where people with knowledge over the different aspects involved in the decision could argue which is the best course of action. Alternatively, another way to provide additional human insight, is to directly use the contributions from employees as an additional source of information, instead of involving them into the discussion.

The fourth aspect suffering changes is the **interactivity** between the user and the BI system. This aspect, related to the previous one, revolves around the ability to interchange information between BI users, as well as contributing with information enriched or created by the own user. This aspect has been emphasized by the Web 2.0 and the new capabilities provided by this interaction, specially on blogs and web applications. In this way, data provided by the BI system should no longer be shown as read-only reports, graphs and figures, but rather as a read and write element in which users can include annotations, link with other elements, and easily send the information to other BI users. This last feature is very relevant, since a recent study from Penteo [41] showed that, nowadays, Excel spreadsheets are the third most used BI tool. The main reason behind this trend is that executives find easy to interact with it, and because is easy to send a spreadsheet to a partner which he can open right away.

The fifth aspect is **how data is presented**. Currently, most BI systems are focused on just showing data in different ways like bar graphs, spreadsheets, aggregations, etc. However, these presentations just focus on how to show the information provided by the data, and do not take into account the cognitive model of the user, or the pursued strategies in order to improve the business performance. Therefore, it is harder for BI users to actually find relevant patterns or develop new strategies using this visualizations. In order to solve this issue, different alternatives have been proposed. Some of these alternatives are traditionally introduced in BI tools, such as linking the data with balanced scorecards, while other proposals also point to link goal models, or even with business process models, with the stored data, thus easily identifying which parts of the business strategy require improvement.

Finally, the last aspect, is related to the **focus of analysis**. Traditionally, there has been an extensive use of descriptive data mining techniques [42]. These techniques are used for example, to segment groups of clients and identifying different client profiles, analyze past trends or problems, and describe the evolution of a certain business activity, like sales. Now, this focus is shifting

towards predictive analysis, centered on supporting decisions like "will there be enough products to meet the demand? should I produce more?". However, since the environment is changing at a very fast rate, these questions now refer to the inmediate future, introducing a strong time restriction on the answer.

After having presented the different aspects of BI which are experiencing an evolution, we will proceed to present the technical challenges which must be overcomed in order to support the previously described features.

5.5 Technical Challenges of Business Intelligence 2.0

The new aspects to be introduced in BI 2.0 cannot be accomplished unless the necessary technology is developed. Therefore, in order to reach the new envisioned BI, a series of technical challenges must be overcomed.

- **Real-time Data warehouses [5]:** In order to achieve a real-time flow of data and be able to process it, the way of capturing this data must be modified. Traditionally, data is captured into transactional databases when a transaction related to a business process ocurrs. Then, at some point determined by the refreshing cycle of the DW, the new data is cleaned, processed, and loaded through ETL processes (*bulk* feed). Afterwards, data can be used for analysis and supporting decisions. However, this process is time consuming and has a negative impact on the performance of both transactional systems (from where data is obtained) and decisional systems (since ETL processes perform operations on tables in the DW). Therefore, this approach does not meet the real-time requirements of BI 2.0, rendering unable to use the decisional support systems while they are being loaded with new data. In order to overcome this technical challenge, two main modifications must be performed on ETL processes. On the one hand, data must be captured as it is being recorded in transactional systems (*trickle* feed). This can be achieved through triggers in transactional databases, logs, and other existing techniques. On the other hand, the whole ETL process should be modelled and automated, supporting the incoming data as a workflow (see figure 5.3) and considering potential exceptions. Therefore, data being loaded might not be complete, and could have information about certain dimensions of the DW missing, or even more, it could present an unexpected pattern, making it necessary to correctly define exception management policies. Once the delay introduced by ETL processes has been minimized, query delay can be minimized using parallelization, hardware acceleration and main-memory database approaches, in order to obtain a seamless real-time BI process. Finally, the last aspect to consider in real-time DWs is the analysis process. Since data is constantly flowing into the DW, the information stored is being modified constantly. Therefore, users are not able to perform multiple analysis over the same set of data, since when they perform a second analysis, the data has already changed, producing inconsistent

results. In order to overcome this limitation, the current approach is to save snapshots of the underlying data, allowing the decision makers to analyze the snapshots multiple times.

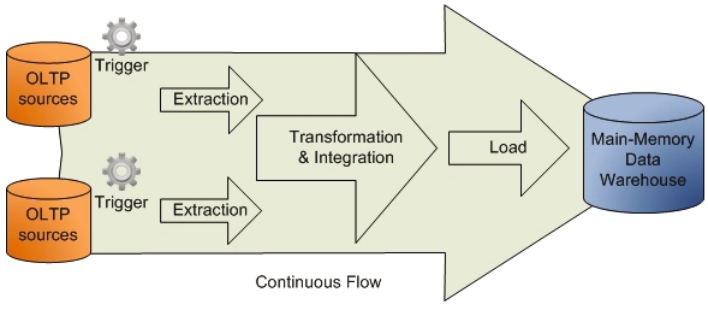


Fig. 5.3. Real time ETL processes with a continuous parallel workflow

- **Scalability:** Scalability issues can affect many different parts of the decision-making process. From the transactional databases, from where the information is being extracted, to the DW, used as data source for the BI solution, or even the network providing the results to the different users. Since in the envisaged BI 2.0 the number of BI users is expected to increase, scalability issues have an increasing importance. There are three main approaches for solving these issues. The first one is by providing more powerful single elements of hardware, while the second is simply providing more hardware elements. The third one is a combination of the previous two, while trying to achieve the most cost-effective solution. In the first group, we can see more and more powerful DW servers being developed by different companies. Some examples are the Exadata from Oracle [43] and the DW server architecture from Netezza [44]. These DW servers include very powerful characteristics with a high number of cores and huge RAM size, in order to increase the speed of the queries formulated against the database. In the second group, we have the recently seen cloud computing services, which were introduced in Section 5.3. Examples of cloud computing services are the Amazon services (s3, ec2), Azure Cloud from Microsoft, and the recent iCloud from Apple. The first solution has the advantage that the information is stored and hosted by the organization, which potentially can provide more privacy and security. On the other hand, cloud services are more flexible to meet the fluctuations in demand throughout the day without incurring in unnecessary costs, and hide some of the technical problems from the users, as described in [27].
- **Semi-structured and unstructured data:** Nowadays clients interchange opinions through social networks and comments posted in forums,

blogs, etc. while competitors post special offers on their website. This means that the business no longer has all the relevant and necessary information stored [23]. Therefore, all this information must be gathered from the outside. This information is highly relevant for the business. Knowing the opinions and ratings given by clients about certain products [32], or the marketing strategies used by competitors, can provide an edge and allow to quickly react by offering special counter promotions. However, this information is either presented in a semi-structured format (as XML files) or, in most cases, in unstructured natural language or as content inside a webpage. Therefore, and due to the size of information which must be analyzed, it must be automatically parsed by using special algorithms and natural language processing (NLP) tools, in order to identify the relevant information and be able to link it with the information stored by the organization. However, due to complexity and ambiguity of the natural language, NLP tools currently are not accurate enough to guarantee that the information obtained is correct. Therefore it is recommended to not directly integrate this information with the one gathered from operational systems and business processes, as it would lower the quality of the data and potentially harm decisions taken. This area is still open for further research.

- **Predictive data mining:** As the focus of analysis is shifted towards the inmediate future, the importance of predictive analysis increases. Since the amount of information to be analyzed in order to take a decision is too big for a human expert, this analysis must be done automatically. Therefore, it is necessary to develop algorithms which analyze the data and describe the current trend and the expected evolution, in order to support the new information needs. Furthermore, it is also important to lower the technical knowledge required to apply these algorithms, as decision makers have been making little use of data mining techniques in the past [45]. However, although there are already a number of predictive data mining techniques [46], the most important restriction introduced in BI 2.0 is the time constraint. If the result can not be obtained in the required space of time, then the result is rendered useless. Using the example from the previous section, if our algorithm predicts that we will need additional supply to meet the demand for the rest of the day but the result is obtained the next morning, then nothing can be done to avoid the situation. In this way, predictive techniques used must obtain results in delimited periods of time, even if some precision must be lost. This area is also open for further research.
- **Analyzing business processes:** Recently, business processes have gained increased attention from the community, and more and more research is being conducted in this area [47], specially focused on business process models. Business process models describe the flow of data through a specific business activity, detailing how the state of data objects evolves throughout the process. Research in this area has dealt with identifying

deadlocks inside processes, providing levels of abstraction to make them understandable, or even obtaining business process models from natural language specifications [48]. However, business process models lack information about the structure of the underlying data, as well as the relationship between the business process and the business strategy. Therefore, it is necessary to integrate this models with the stored data and the business goals, in order to be able to analyze how anomalies, errors and improvements in business processes affect the business performance. These aspects are related to Business Process Intelligence, and is an area open for further research.

- **Linking data[30]:** The most important feature about a specific piece of data in order to analyze it is the different relationships it has with other data. Some of these relationships are already explicitly included in data stored in the enterprise DW (i.e. facts, dimensions and hierarchies). Other relationships are implicit, and can be discovered through the use of data mining techniques, obtaining for example, a set of rules implicitly present in the data. However, some relationships relate two independent sets of data, and might only be identified by a human analyst while inspecting information from different sources (e.g. an analyst comparing the sales of a product and visualizing similar products from competitors). This is specially interesting when trying to analyze data obtained from the Web together with the existing information inside the organization. In order to be able to automatically reason and infer new knowledge from these relationships, they must be modelled and semantically tagged, allowing to differentiate the kind of relationship established. In order to model these relationships, an approach to link the data must be developed. Some examples of proposals for linking data include traceability approaches [22] or using ontologies and semantic tagging [31]. While generic approaches can be applied to link the data, the semantics of the relationships are typically domain dependent, since an exhaustive definition of every possible relationship between two pieces of data would be non-viable. Since this area of research is growing, and recently new sets of relationships have appeared (like those in social networks), it is open for further research.

After having presented the different technical challenges of BI 2.0, we will present some tools from different vendors and will analyze which features are already integrated and which ones still require further effort to be included.

5.6 General Overview of Tools Stepping towards BI 2.0

As BI has gained more and more attention, different vendors have developed tools for querying the underlying information stored at enterprise DWs. Some of these vendors are big companies like Oracle, Microsoft, SAP, IBM, Microstrategy or SAS, while others are open source groups like the one behind Pentaho BI. In this section, we take a brief look at different tools provided

by some of these vendors and how they are integrating the characteristics we have described in Section 5.3.

The first tool we analyze is the BI tool from MicroStrategy [39]. Microstrategy includes a web interfaces for their BI tool, which is most important in Mobile BI. The Mobile BI application is designed for iPad, Blackberry and iPhone devices, and provides a way to analyze the stored data from anywhere. On the other hand, Microstrategy also provides scorecards and dashboards to connect the underlying data and Key Performance Indicators with the business strategy and strategic goals, reducing the cognitive effort of the users. Finally, Microstrategy is providing some support for predictive analysis through mathematical functions, in order to meet the new analysis requirements. On the negative side, the BI tool lacks predictive data mining support with algorithms, some interactivity and collaboration between BI users, and integration of business processes.

The second tool we analyze is Pentaho [40]. Pentaho is an open source BI tool, which can be integrated with different platforms and includes a special enterprise version. Pentaho provides a web interface, as MicroStrategy, as well as a dashboard for linking data, including support for scorecards but requiring some effort to integrate them. Pentaho also includes support for predictive analysis, including some data mining algorithms. In the collaborative aspect, some work has been done to integrate LifeRay and Pentaho in order to provide a social feeling. On the negative side, Pentaho also lacks support for integrating business processes as well as interactivity.

The third tool we analyze is Cognos from IBM [37]. Cognos, as the previous two tools, includes a web interface, dashboards and scorecards for linking data. Moreover, Cognos also includes support for mobile devices, as well as collaborative BI support, allowing to include annotations, opinions, and enriching and sharing data, in order to achieve a collaborative decision process and using the collective intelligence. On the negative side, the predictive capabilities of Cognos are limited and require to use additional software for fully dealing with predictive analysis.

The fourth tool we analyze is SAS BI [49]. SAS provides a web interface with highly customizable dashboards, which can be linked with other elements and dashboards. Along with interactive widgets and visualization, SAS BI includes special visualization tools which help to understand the underlying data. SAS also provides support for mobile devices, allowing to analyze the performance indicators and other information from anywhere. On the negative side, there is no information about the predictive data mining support, or the inclusion of collaborative BI.

The fifth tool we analyze is Microsoft BI [38]. Microsoft BI is composed of a series of different technologies, combining Excel, PowerPivot, Sharepoint server, and SQL Server, in order to provide a BI system for the business. As other vendors, Microsoft includes scorecards and dasboard mashups in a web interface through Sharepoint, in order to link data and provide insightful information about the business performance. Moreover, Sharepoint allows to

add tags to our own profile and check the profile of other people, enabling some degree of collaboration. The analysis of detailed data is done through Excel and PowerPivot, which allows to also include data from webpages. On the negative side, since this BI system is composed of several technologies, some expertise is required to use them together without a loss of time derivated from integrating the results. On the other hand, the predictive data mining techniques provided are limited to Excel functions (aside from using algorithms directly on the database which stores the information), and the collaborative part lacks some features like enriching the data so other users can see these anotations, as well as an easy way to interact while analyzing the data.

The last tool we analyze is SAP BI [50]. SAP BI is composed, as in the case of Microsoft, of many different tools. SAP includes complete support for analyzing the business strategy with a combination of desktop and web applications. Among the features provided, SAP includes dashboards and scorecards, a web interface for ad-hoc analysis, support for mobile devices, and a complete workbench for data mining tasks. Furthermore, SAP also includes integration capabilities with Excel and enterprise applications, which empowers the analysis capabilities. On the negative side, although the analysis capabilities are very powerful, the BI system currently lacks support for active collaboration and enriching information (although this aspect is being improved [6]), which limits the applicability of collaborative BI and crowdsourcing.

Table 5.1. Summary of features provided by BI tools from different vendors

Tool	MicroSt	Pentaho	Cognos	SAS	Microsoft	SAP
Web interface	Y	Y	Y	Y	Mixed	Mixed
Scorecards	Y	Requires effort	Y	Y	Y	Y
Dashboards	Y	Y	Y	Y	Y	Y
Interactivity	N	N	Y	Y	Limited	-
Collaboration between users	N	LifeRay	Y	-	Some	Under develop.
Enriching information	N	N	Y	N	Some	Under develop.
Predictive Analysis	Math functions	Weka	Additional software	-	Excel	Complete support
Business Processes	N	N	N	N	N	Y
Integration	Single platform	Independent modules	Single platform	Single platform	Multiple software	Multiple software
Additional features	Dedicated Mobile BI	Open source BI	-	Visualization tools	Supports web data	Enterprise integration

Most of the tools support the use of cloud computing through the SaaS approach, in order to deal with the scalability issues. In this way, BI services are provided as a service from the cloud, where the BI server is deployed. However, some tools also include specific support for cloud computing, like Cognos (guide to deploy the server in the cloud), Microsoft BI (Azure cloud), and Pentaho (Hadoop, for data integration). On the other hand, in most tools, data is linked by means of dashboards and additional code, rather than interaction. While this enables a way to present the data in a linked way, it requires certain effort to achieve and is not really intuitive. Moreover, the presentation of the data can be connected to business strategic goals in some tools but it is not used as the main way of visualizing the existing data, which is still data-driven. Finally, most tools lack interactivity to enrich the data and contribute to a collaborative BI, by providing means to easily interchange information and opinions with other BI users.

5.7 Conclusions

In this paper, we have presented an overview of the different aspects related to BI 2.0. We have described each aspect and its motivation, as well as the technical challenges which have to be overcomed. The main consensus about BI 2.0 is that it should be processed in real-time, in a more intuitive and collaborative manner, as opposed to the previous generation. Some of the technical challenges already have the necessary technology to be overcomed, but still require effort to achieve its integration into BI tools. Most of these tools already include a few envisioned characteristics at the moment. However, we have not yet reached the envisioned BI 2.0, and some time will be required for most tools to completely adapt and include the required features.

On the other hand, further research is necessary to identify the most effective way of presenting data, as well as developing easy to use, more accurate, and faster predictive algorithms, while also identifying which are the best practices for interacting and contributing with information inside the business environment.

Acknowledgments. This work has been partially supported by the MESOLAP (TIN2010-14860) and SERENIDAD (PEII-11-0327-7035) projects from the Spanish Ministry of Education and the Junta de Comunidades de Castilla La Mancha. Alejandro Maté is funded by the Generalitat Valenciana under an ACIF grant (ACIF/2010/298). We would like to thank our colleagues, Marta Zorrilla, Oscar Fernández, José Norberto, Irene Garrigós, and Florian Daniel, for editing with us the book "Business Intelligence Applications and the Web: Models, Systems and Technologies". We would also like to thank all the authors who contribute with different chapters in this book sharing their experience.

References

1. Gartner Group: Gartner Group BI Revenue Analysis (2009), http://www.gartner.com/it/page.jsp?id=1357514
2. Zorrilla, M., Mazón, J.N., Garrigós, I., Daniel, F., Trujillo, J. (eds.): Business Intelligence Applications and the Web: Models, Systems and Technologies. IGI Global (2011)
3. Bhowmick, S., Madria, S., Ng, W.K., Lim, E.: Web warehousing: Design and issues. In: Kambayashi, Y., Lee, D.-L., Lim, E.-p., Mohania, M., Masunaga, Y. (eds.) ER Workshops 1998. LNCS, vol. 1552, pp. 93–105. Springer, Heidelberg (1999)
4. Essaidi, M., Osmani, A.: Business Intelligence-as-a-Service: Studying the Functional and the Technical Architectures [2]
5. Thiele, M., Lehner, W.: Real-Time BI and Situational Analysis [2]
6. Berthold, H., Rösch, P., Zöller, S., Wortmann, F., Carenini, A., Campbell, S.: Towards Ad-hoc and Collaborative Business Intelligence [2]
7. Greg, N.: Business Intelligence 2.0: Are we there yet?, http://support.sas.com/resources/papers/proceedings10/040-2010.pdf
8. 2nd International Workshop on Business intelligencE and the WEB, http://gplsi.dlsi.ua.es/congresos/beweb11/
9. International Workshop on Business intelligencE and the WEB, http://gplsi.dlsi.ua.es/congresos/beweb10/
10. Golfarelli, M., Mandreolib, F., Penzoa, W., Rizzi, S., Turricchia, E.: BIN: Business Intelligence Networks [2]
11. Kimball, R.: The data warehouse toolkit. Wiley-India (2009)
12. Inmon, W.H.: Building the data warehouse. Wiley-India (2009)
13. Giorgini, P., Rizzi, S., Garzetti, M.: GRAnD: A goal-oriented approach to requirement analysis in data warehouses. Decision Support Systems 45(1), 4–21 (2008)
14. Abelló, A., Samos, J., Saltor, F.: YAM2: a multidimensional conceptual model extending UML. Information Systems 31(6), 541–567 (2006)
15. Luján-Mora, S., Trujillo, J., Song, I.-Y.: A UML profile for multidimensional modeling in data warehouses. Data & Knowledge Engineering 59(3), 725–769 (2006)
16. Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R model for the multidimensional paradigm. Advances in Database Technologies (2004)
17. Tryfona, N., Busborg, F., Borch Christiansen, J.: starER: A conceptual model for data warehouse design. In: Proceedings of the 2nd ACM International Workshop on Data warehousing and OLAP, pp. 3–8. ACM (1999)
18. Mazón, J.N., Pardillo, J., Trujillo, J.: A Model-Driven Goal-Oriented Requirement Engineering Approach For Data Warehouses. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 56–71. Springer, Heidelberg (2007)
19. Yu, E.S.K.: Modelling strategic relationships for process reengineering. PhD thesis, University of Toronto, Toronto, Ont, Canada (1995)
20. Mazón, J.-N., Trujillo, J.: An MDA approach for the development of data warehouses. Decision Support Systems 45(1), 41–58 (2008)
21. Mazón, J., Trujillo, J.: A hybrid model driven development framework for the multidimensional modeling of data warehouses. ACM SIGMOD Record 38(2), 12–17 (2009)

22. Maté, A., Trujillo, J.: A Trace Metamodel Proposal Based on the Model Driven Architecture Framework for the Traceability of User Requirements in Data Warehouses. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 123–137. Springer, Heidelberg (2011)
23. Betanyeb, F., Maïz, N., Mahboubi, H., Favre, C., Loudcher, S., Harbi, N., Bousaïd, O., Darmont, J.: Innovative Approaches for Efficiently Warehousing Complex Data from the Web [2]
24. Ferrández, A., Peral, J.: The benefits of the interaction between data warehouses and question answering. In: Proceedings of the 2010 EDBT/ICDT Workshops, p. 15. ACM (2010)
25. Marotta, A., González, L., Etcheverry, L., Rienzi, B., Ruggia, R., Serra, F., Martirena, E.: Quality Management in Web Warehouses [2]
26. Rizzi, S.: New Frontiers in Business Intelligence: Distribution and Personalization. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 23–30. Springer, Heidelberg (2010)
27. Armbrust, M., et al.: Above the clouds: A berkeley view of cloud computing. Technical report, EECS Department, University of California, Berkeley, UCB/EECS-2009-28 (February 2009)
28. Gruber, T.: Collective knowledge systems: Where the social web meets the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 6(1), 4–13 (2008)
29. Howe, J.: Crowdsourcing: Why the power of the crowd is driving the future of business. Crown Business (2009)
30. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on the Semantic Web and Information Systems 5(3), 1–22 (2009)
31. Berlanga, R., Romero, O., Simitsis, A., Nebot, V., Pedersen, T.B., Abelló, A., Aramburu, M.J.: Semantic Web Technologies for Business Intelligence [2]
32. Balahur, A., Boldrini, E., Montoyo, A., Mártinez-Barco, P.: OpAL: a System for Mining Opinion from Text for Business Applications [2]
33. Golfarelli, M., Rizzi, S., Cella, I.: Beyond data warehousing: what's next in business intelligence? In: Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, pp. 1–6. ACM (2004)
34. O'Reilly, T.: What is Web 2.0: Design patterns and business models for the next generation of software. Communications and Strategies 65, 17 (2007)
35. Byung-Kwon, P., Song, I.Y.: Incorporating Text OLAP in Business Intelligence [2]
36. Pallotta, V., Vrieling, L., Delmonte, R.: Interaction Business Analytics [2]
37. IBM: Cognos Business Intelligence (June 2011)
38. Microsoft: Microsoft Business Intelligence (June 2011)
39. Microstrategy: MicroStrategy Business Intelligence (June 2011)
40. Pentaho: Pentaho Business Intelligence (June 2011)
41. Penteo: Tendencias en el uso de BI en España (June 2011)
42. Berkhin, P.: Survey of clustering data mining techniques. Grouping Multidimensional Data: Recent Advances in Clustering, pp. 25–71 (2006)
43. Oracle: A Technical Overview of the Oracle Exadata Database Machine and Exadata Storage Server. White Paper (April 2011)
44. IBM: The Netezza Data Appliance Architecture: A Platform for High Performance Data Warehousing and Analytics. White Paper (January 2011)
45. Pedersen, T.B.: How Is BI Used in Industry?: Report from a Knowledge Exchange Network. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 179–188. Springer, Heidelberg (2004)

46. Weiss, S., Indurkhya, N.: Predictive data mining: a practical guide. Morgan Kaufmann (1998)
47. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.: Business process intelligence. Computers in Industry 53(3), 321–343 (2004)
48. Friedrich, F., Mendling, J., Puhlmann, F.: Process Model Generation from Natural Language Text. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011)
49. SAS: SAS Business Intelligence (June 2011)
50. SAP AG: SAP Business Intelligence (June 2011)

6

Graph Mining and Communities Detection

Etienne Cuvelier and Marie-Aude Aufaure

MAS Laboratory, Business Intelligence Team
Ecole Centrale Paris,
Châtenay-Malabry, France
{etienne.cuvelier,marie-aude.aufaure}@ecp.fr

Summary. The incredible rising of on-line social networks gives a new and very strong interest to the set of techniques developed since several decades to mining graphs and social networks. In particular, community detection methods can bring very valuable informations about the structure of an existing social network in the Business Intelligence framework. In this chapter we give a large view, firstly of what could be a community in a social network, and then we list the most popular techniques to detect such communities. Some of these techniques were particularly developed in the SNA context, while other are adaptations of classical clustering techniques. We have sorted them in following an increasing complexity order, because with very big graphs the complexity can be decisive for the choice of an algorithm.

Keywords: Graph Mining, Community Detection, Data Mining.

6.1 Introduction

In the actual interconnected world, and the rising of online social networks the graph mining and the community detection become completely up-to-date. If the interest for social networks analysis exists, even in germ, from the beginning of sociology, in the works of Emile Durkheim or Auguste Comte, a more systematic study started in the 1930s with the work of Moreno, Davis and Mayo (see [1] and [2] for a more complete state of the art of the social network analysis). And this systematic study includes the use of graph theory. The theory of graphs exists since Euler's solution of the Knigsberg's bridges problem in 1736, but networks, treated in the pioneer times of Euler (for graph theory) and Moreno (for social network analysis), contained only some dozens nodes at most, while the rising of computer science and one of its fields, the data analysis (see [3] or [4] for a good introduction this latter), allow to analyze graphs of big sizes, and permit to develop one of the task of social network analysis: finding groups with high concentrations of relations within the group and low concentration between these groups, which is called community detection [5, 6, 7].

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 117–138, 2012.

This overview of the community detection algorithms is organized as follows: in Section 2 we introduce the basics of social network and network theory. In Section 3 we give a review of the definition of a cluster or community in social network analysis. Then, in Section 4 we take a look to the set of measures which can be used in clustering of social networks. Sections 5, 6, 7 and 8 give a review of the most important clustering methods. The order chosen for this review is directly linked to the complexity : we begin with the "cheapest", and then more scalable (partitional and hierarchical methods), then we continue with a more greedy but very efficient technique : the spectral clustering, and we end with the Galois lattices which are costly methods, but with very rich results. Then we close this part with a discussion.

6.2 Social Networks

Even if Moreno [8] was the first to use points and lines to represent social configurations, it was Cartwright and Harary [9] which made the link with the graph theory, and thus introduced the actual graph representation of social network (see [10] for a review of the evolution of social networks representation): individuals are represented using points, called *nodes* or *vertices*, and social relationships are represented using lines, called *edges* or *links*, between nodes. In figure 6.1 we show an example of social networks using this graph

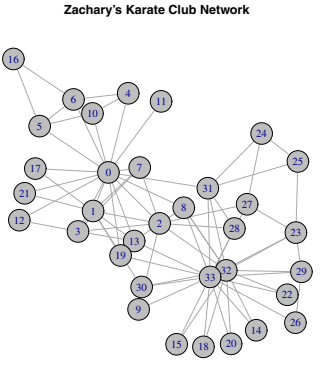


Fig. 6.1. A classical example of social network : the Zachary's Karate Club

representation: The first one (fig. 6.1) is a the well-known graph [11]: the Zachary's Karate Club, it consists in 34 vertices, the members of a karate club in the United States, who were observed during a period of three years. Edges connect individuals who were observed to interact outside the activities of the club.

Let use introduce here the basic concepts of graph theory. A *graph* G is defined as a pair of sets: $G = (V, E)$, where V is the set of *vertices* or *nodes*, and E the set of *edges* or *links* which connect vertices. The two vertices connected by an edge are called *endpoints* of this latter. Conversely the edges connecting a vertice to the other vertices are called *incident* edges of the node. The number of vertices $|V| = n$ is called the *order* the graph, while the number of edges $|E| = m$ is called the *size* of the graph. If $|E| = n(n-1)/2$, i.e. if any pair of vertices are connected, then the graph is called *complete* .

A graph can be *directed* or *undirected*. In the first case, also called *digraph*, an edge is denoted as pair (v, w), where v is the origin and w the target, and in the social networks framework it means: "v is in relation with w", the opposite being true if and only if there exists an edge (w, v). If the graph G is undirected, then an edge is denoted as the set of its vertices : $\{v, w\}$. For the sake of simplification, when we will give a definition valid in both cases, we will use this last notation.

Most of the times, vertices are labeled, but it can also be the case for edges, then G is called a *labeled graph*. Moreover, if exists a function $\omega : E \to \mathbb{R}$ that assigns a weight for each edge, then G is a *weighted* graph.

Two vertices v and u are called *neighbors* or *adjacent*, if they are connected by an edge. The set of neighbors of a node v, denoted $\Gamma(v)$, is called its *neighborhood*.

The topology of the graph can be captured in the *adjacency matrix* $A = (a_{i,j})$: where

$$a_{i,j} = \begin{cases} 1 \text{ if } (v_i, v_j) \in E, \\ 0 \text{ otherwise.} \end{cases} \tag{6.1}$$

If G is a weighted graph, then $a_{i,j} = \omega(v_i, v_j)$ and in this case we prefer to use the notation $W = (w_{i,j}) = (\omega(v_i, v_j))$. Of course in an unweighted graph $\omega(v_i, v_j) \in \{0, 1\}$. For an unweighted graph, the *degree* of a vertice v, denoted $deg(v)$, is defined as the number of incident edges, but a more general definition is given using the weights matrix W:

$$deg(v_i) = \sum_{j=1}^{n} w_{i,j}. \tag{6.2}$$

A *subgraph* $G' = (V', E')$ of a graph $G = (V, E)$ is such $V' \subset V$, $E' \subset E$ and $\{v, u\} \in E'$ implies $v, u \in V'$. The graph G is a *supergraph* of G'. A subset C of V can define an *induced subgraph* $G(C) = (C, E(C))$, where

$$E(C) = \{(v, u) \in E | v, u \in C\} \tag{6.3}$$

A complete subgraph is called a *clique*.

The *density* of a subgraph $C(V(C), E(C))$ is the ratio between $|E(C)|$ and the maximum possible number of edges:

$$\delta(G(C)) = \frac{|E(C)|}{|V(C)|(|V(C)| - 1)/2} \tag{6.4}$$

this definition still being valid for the whole graph G.

A partition of the vertices set V in two subsets C and $V \setminus C$ is called a *cut*. The *cut size*, denoted $c(C, V \setminus C)$ is the number of edges of G joining vertices of C with vertices of $V \setminus C$:

$$c(C, V \setminus C) = |\{\{u, v\} \in E | u \in C, v \in V \setminus C\}| \tag{6.5}$$

Both set C and $V \setminus C$ define the cut, but usually the cut is identified by the smaller one.

A *path* between two vertices v and u in a graph G is a sequence of edges starting with $v_0 = v$ an such the last one is $v_k = u$:

$$\{v, v_1\}, \{v_1, v_2\}, \cdots, \{v_{k-1}, u\}. \tag{6.6}$$

A path P is also a subgraph of G: $P = (V(P), E(P))$ with $V(P) = \{v_0, \cdots, v_{k1}\}$ and $E(P) = \{\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{k-1}, v_k\}\}$. The *length* of a path is the number of edge in this path, k in expression (6.6). A *shortest path* or *geodesic* between two vertices is a path of minimal length, and the distance between two vertices is the length of a geodesic between these two vertices. The *diameter* of a graph is the maximal distance that can be found between two nodes.

If no vertice is repeated, then the path is *simple*. If there exists a path between two vertices v and u, they are *connected*. The graph is also called a *connected graph* if for any pair of vertices v and u, there is, at least, one path connecting v and u. Otherwise, i.e. if there is some vertices which cannot be reached from other, then the graph is *disconnected*. If there is two nodes without path between them, then there is, at least two connected subgraphs, and a maximal connected subgraph is called a *connected component*. The *edge connectivity* of a graph G is the minimal number of nodes to be removed so that G is disconnected, and is denoted $k(G)$.

A *cycle* is a path such the first and the last node are equal, i.e. $v_0 = v_k$. A graph without cycle is called a *forest* or an *acyclic graph*, and a connected forest is a *tree*. The edge connectivity $k(T)$ of a tree T is equal to one, because, if the tree contains n vertices, it had $n-1$ edges, and if any of these is removed, the tree is divided in two disconnected trees.

A connected acyclic subgraph $G' = (V', E')$ such $V' = V$, i.e. all the vertices of G are also in G', is called a *spanning tree*. Every connected graph contains a least a spanning tree. For weighted graphs, a *minimum spanning tree* is the spanning tree such the sum of the weights of the edges is minimal. We can define a *maximum spanning tree* similarly.

6.3 Community Definitions

In the clustering framework a *community* is a cluster of nodes in a graph [12], but a very important question is *what is a cluster?* Even in the clustering literature there is non complete agreement between all authors (see [13] for a

review), but most of the time, the objects in a cluster must be more similar than objects out of this cluster: *the objects are clustered or grouped based on the principle of maximizing the intra-class similarity and minimizing the inter-class similarity.* Let us remark that, this definition implies the necessity to define the notions of similarity measure and/or cluster fitness measure.

In the graph framework, we can agree that the goal of clustering is to divide the data set into clusters, such nodes of a cluster must be more connected with nodes of this cluster, than with nodes outside of the cluster [7] and [6]. It implies that it must exists at least a path between two nodes of a cluster, and this path must be internal to the cluster.

Then the transposition of the above definition of a cluster into the graph context could be the following: *the vertices are clustered or grouped based on the principle of maximizing the intra-class connections and minimizing the inter-class connections.*

There is several manner to quantify these internal and external connections of a cluster C. A first one is to divide the degree of a vertice in two parts: the internal degree $deg_i(v, C)$ and the external degree $deg_e(v, C)$:

$$deg_i(v, C) = |\Gamma(v) \cap C| \tag{6.7}$$

$$deg_e(v, C) = |\Gamma(v) \cap (V \setminus C)|. \tag{6.8}$$

And, as $\Gamma(v) = \{\Gamma(v) \cap C\} \cup \{\Gamma(v) \cap (V \setminus C)\}$

$$deg(v) = deg_i(v, C) + deg_e(v, C). \tag{6.9}$$

Of course, if $deg_e(v, C) = 0$, then $v \in C$ is surely a good assignation for v, and conversely if $deg_i(v, C) = 0$, then we must have $v \notin C$.

The internal and external degrees, can be seen as the "vertice's point of view" of indicators of the belonging to a cluster. For a "cluster's point of view", we must take a look at the notion of graph density. The *intra-cluster density* $\delta_i(C)$ and the *inter-cluster density* $\delta_e(C)$ are adapted versions of the density of a subgraph defined by expression (6.4), the first one being the quotient of the number of internal edges of C and the maximal possible number of internal nodes:

$$\delta_i(C) = \frac{|\{\{u, v\} | u, v \in C\}|}{|C|(|C| - 1)/2} \tag{6.10}$$

and the second one being the result of the number of edges with one vertice inside C, and the other outside of C, divided by the maximal possible number of edges in this configuration:

$$\delta_e(C) = \frac{|\{\{u, v\} | u \in C, v \notin C\}|}{|C|(|G| - |C|)}. \tag{6.11}$$

Of course for a given cluster C we expect $\delta_i(C)$ and $\delta_e(C)$ to be substantially, respectively, larger and smaller than the average density $\delta(G)$. And, in a

"partition's point of view", the internal density of the partition, given by the sum of densities over all the clusters, must be appreciably higher than the density of the graph $\delta(G)$.

The comparison of the density of inner ties versus the average density of the graph, leads to define a community in comparison to the rest of the graph, but a community can be considered independently of the graph as a whole. Local definitions of communities [6], focus only on the cohesion of the studied subgraph, including possibly its direct neighborhood, but ignoring the rest of the graph. In [2] Wasserman identify four criteria to define a cohesive subgroup: *complete mutuality*, *reachability*, *nodal degree*, *internal versus external cohesion*.

The concept of *complete mutuality* states that, in a very strict sense, in a community, all member of a subgroup must be "friends" with all members of the subgroup. In graph theory, it corresponds to a clique [14].

But the definition of a community as a clique is very too strict that is why relaxed definitions of the notion of clique leads to the *reachability*. An *n-clique* (or *k-clique*) [15] is a maximal subgraph such, for any pair of vertices, there exists at least a geodesic no larger than n (or k). The classical clique is then a 1-clique. But a geodesic path of an n-clique could run outside of this latter, and then the diameter of the subgraph can exceed n. That is why the notion of *n-club* and *n-clan* was suggested [16]. An *n-clan* is an *n-clique* with diameter not larger than n, while an *n-club* is a maximal subgraph of diameter n.

The use of the *nodal degree* to define a community imposes a constraint on the number of adjacent vertices. A *k-plex* [17] is a maximal subgraph such each vertice is adjacent to all other vertices of the subgraph except for k of them. Conversely a *k-core* is a maximal subgraph such each vertice is adjacent to, at least, k other vertices of the subgraph [18].

Finally, comparing *internal versus external cohesion*, an *LS-set* [19], or *strong community* [20] is a subgraph C such for each node $v \in C$ we have $deg_i(v, C) > deg_e(v, C)$.

Another point of view to define a community is to say that the number of ties with the outside of the community must be low. That leads to use the notion cut size to define a cluster, and to try to minimize it. Rather than using directly the cut size, the conductance [21] $\Phi(C)$ of a community C is defined to taking into account the order of the cluster and the outside of the cluster:

$$\Phi(C) = \frac{c(C, V \setminus C)}{\min\{deg(C), deg(V \setminus C)\}} \tag{6.12}$$

where $deg(C)$ and $deg(V \setminus C)$ are the total degrees of C and of the rest of the graph. The minimum of the conductance is obtained when a C community have a low cut size and when the total degree of the cluster and its complement are equal [22].

The cut size is sometime also called the *external degree* of a community C

$$deg_e(C) = |\{\{u,v\} \in E | u \in S, v \in V \setminus S\}| \tag{6.13}$$

while the *internal degree* is defined by

$$deg_i(C) = |\{\{u,v\} \in E | u,v \in S\}| \tag{6.14}$$

and then $deg(C) = deg_i(C) + deg_e(C)$. Internal and external degrees are used to define the *relative density* [23]:

$$\rho(C) = \frac{deg_i(C)}{deg(C)}. \tag{6.15}$$

A community with strong inner ties must have a higher ρ.

Another idea to define a community, is given using the possible flow of information. In a group, given two nodes, at least one shortest path between these nodes, passing through the edges of the group must exist. Conversely if an edge is on many shortest path between several nodes, we can suppose that it is a connection edge between two communities. Finding the connecting edges permits to find the connected communities. It is the idea of the *betweenness*, and more precisely the *edge (or site) betweenness* introduced by [24]. The figure 6.2 gives an illustration of a node with a maximum edge betweenness.

Finally, the notion of *edge connectivity* k is also used to define a community: an *highly connected subgraph (HCS)* [25] is a subgraph $C \in G$ such

$$k(C) > \frac{n}{2}. \tag{6.16}$$

6.4 Measure for Clusters

Most of the clustering algorithms are based on one or several measures, to be optimized and/or to be used to compare different cluster affectations. We are going to give here the most popular ones.

If we can embed the graph into a n-dimensional Euclidean space, then we can use the classical distances like *Euclidean distance*, *Manhattan distance* or *cosine similarity*, but this embedding into a n-dimensional space can be seen as an artificial construction, and then a distance defined in such a space, used on a graph, is subject to the same criticism. It can be more suitable to work directly with informations included in the adjacency matrix, defining a distance based on this latter [26], [2]:

$$d_{i,j} = \sqrt{\sum_{k \neq i,j} (a_{i,k} - a_{j,k})^2}. \tag{6.17}$$

This dissimilarity measures the structural equivalence [27]: two vertices are structurally equivalent if they have the same neighbors, which is the case when $d_{i,j} = 0$.

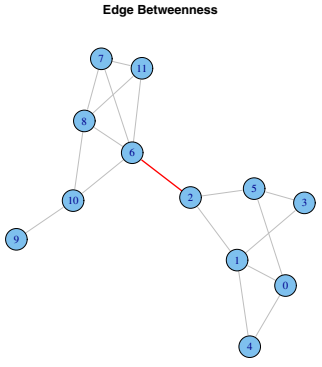


Fig. 6.2. An example of maximum edge betweenness (edge between nodes 2 and 6)

Another measure directly defined on the adjacency matrix is the Pearson correlation matrix computed on rows or columns of A:

$$c_{i,j} = \frac{\sum_{k=1}^{n}(a_{i,k} - \mu_i)(a_{j,k} - \mu_j)}{n\sigma_j\sigma_j} \tag{6.18}$$

with $\mu_i = \sum_k a_{i,k}/n$ and $\sigma_i = \sqrt{\sum_k (a_{i,k} - \mu_k)^2/n}$.

Another popular seed to build (dis)similarity measure is the *Jaccard index* which measures similarity between sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{6.19}$$

A first use of the Jaccard index in the graph theory context is to measure the *overlap* of the neighborhoods of two nodes v and u:

$$\omega(v, u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{|\Gamma(v) \cup \Gamma(u)|} \tag{6.20}$$

which is equal to zero when there is no common neighbors, and one when v and u are structurally equivalent.

6.5 Partitional Clustering

Partitional algorithms try to find a partition of a set of data, with a given number of cluster equal to k. The "best" partition is searched using jointly, most of the times, a distance or dissimilarity measure and a quality criterion of the found partition.

The most popular partitional algorithm (with several variants) is the *k-means clustering* [28]. We give here its most general form. The algorithm try to find a partition $\mathcal{C} = (C_1, \cdots, C_k)$ of a set of n objects denoted $\{x_1, \cdots, x_n\}$, which minimize the *sum-of-square (SSQ)* criterion[1]:

$$g_n(\mathcal{C}) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, c_i) \tag{6.21}$$

where $d(x, c_i)$ measures the dissimilarity between the object x and the *centroid* of the class C_i. Expression (6.21) is called SSQ, because initially $d(x, c_i)$ was computed in the following way : $||x - c_i||^2$.

The general algorithm is the following :

Starting step: Determine an initial vector $(c_1^{(0)} \cdots, c_k^{(0)})$,
Repeat until stationarity: $t \leftarrow t + 1$

Assignment step: Assign each observation to the cluster with the closest centroid:

$$C_i^{(t+1)} = \left\{ x : d(x, c_i^{(t)}) \leq d(x, c_j^{(t)}) \text{ for all } j = 1, \ldots, k \right\} \tag{6.22}$$

Update step: Compute the new centroids $(c_1^{(t+1)}, \cdots, c_k^{(t+1)})$ of the clusters.

Originally centroids of the clusters were computed as the means of the clusters but a more robust version can use the median of the cluster, and [29] have proposed to use as "centroid" (then called prototypes) the most typical subset (pair, triple,...) of objects from this class. Most of the times the starting partition is chosen randomly.

Even if k-means algorithms are efficient, they have several drawbacks

- the choice of k: the number of clusters k is an important input parameter, and an inappropriate choice may leads too non significant results,
- spherical clusters: the algorithm tends to output spherical data, and then works well only when spherical clusters are naturally available in data,
- instability: the random starting partition can leads to a local optimum for the criterion function, and this local optimum can change significantly with another initial partition; to decrease the effect of this instability, a solution it to execute the algorithm several times with different random starting partitions and to retain the solution with the best criterion value.

The complexity of such partitional algorithms is given by $O(m^2 \times n \times k)$ [4] where m is the number of attributes, n the number of objects to cluster and k the number of clusters.

[1] The SSQ is also termed inertia, squared error function, variance criterion, or trace criterion.

6.6 Hierarchical Algorithms

Hierarchical clustering algorithms are divided into two types, depending on whether the partition is refined or coarsened during each iteration:

- *agglomerative algorithms* which start with a set of small initial clusters and iteratively merging these clusters into larger ones,
- *divisive algorithms* which split the dataset iteratively or recursively into smaller and smaller clusters.

At each step, the clustering algorithm musts select the clusters to merge or split by optimizing a quality criterion.

6.6.1 Agglomerative Algorithms

The idea agglomerative algorithms is simple: at the starting point, the n objects to cluster $\{x_1, \cdots, x_n\}$ are their own classes: $\{\{x_1\}, \cdots, \{x_n\}\}$, then at each stage we merged the two more similar clusters.

A dissimilarity measure D between two clusters A and B is mandatory, and for a given dissimilarity measure d between objects, several D exist, we give here the most popular:

- the single linkage: $D(A, B) = \min\{d(x, y) : x \in A,\, y \in B\}$,
- the complete linkage: $D(A, B) = \max\{d(x, y) : x \in A,\, y \in B\}$,
- the average linkage: $D(A, B) = \frac{1}{|A| \cdot |B|} \sum_{x \in A} \sum_{y \in B} d(x, y)$.

In [30] the Lance formula was given to compute $D(C, A \cup B)$ given $D(C, A)$, $D(C, B)$ and $D(A, B)$. The result of the clustering process is shown in a *dendrogram*, like in figure 6.3. In this kind of graph, mergers of communities are drawn like horizontal lines, and the height of this line represents the dissimilarity between the merged clusters (or objects). Thus, in figure 6.3, individuals 24 and 25 are closest than individuals 23 and 27. For a given k it is easy to find a partition of objects, because an horizontal cut of the dendrogram gives one single partition, we have only to choose the corresponding height. In figure 6.3 cuts A, B and C, give respectively 4, 7 and 10 clusters. The dendrogram is a hierarchical structure because each community is fully included in a community of higher level.

The drawbacks of agglomerative hierarchical methods are the following: firstly, vertices of a community may be not correctly classified, and some nodes could be missed even if they have a central role in their cluster [31]; secondly the computational complexity is $O(n^2)$ for the single linkage and $O(n^2 \log n)$ for complete and average linkages, then very big datasets are to avoid. A solution to decrease this computing time is to apply a k-means with a relatively high k, but with $k << n$, and apply the hierarchical clustering on the previous results.

6.6.2 Divisive Algorithms

Betweenness

Using the notion of betweenness [24], it is possible to use a divisive approach to detect communities. The algorithm given in [12] and [5] splits the network into clusters by removing, step after step, edges with the higher betweenness value. In the algorithm, the two following steps are repeated:

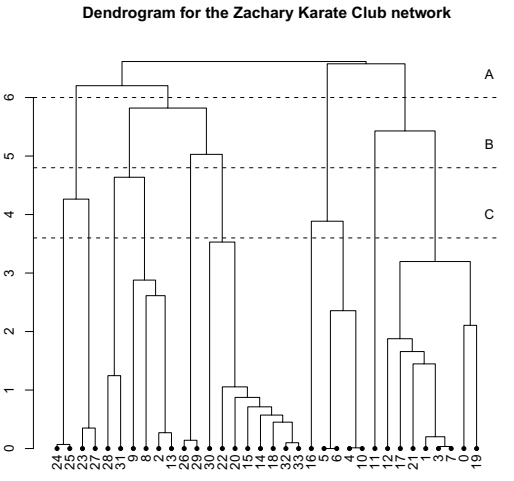


Fig. 6.3. A dendrogram for the Zachary Karate club network

1. compute the edge betweenness for all edges of the running graph,
2. remove the edge with the largest value (which gives the new running graph).

Of course, the immediate question is: when to stop this iterative algorithm? Having a criterion to improve at each step should permit to stop when no improvement is gained with an iteration. Such a criterion is proposed in [31]: the *modularity*. Modularity is normally defined for weighted graphs, but can be applied on unweighted graph using $\omega(u, v) = 1$ for all edges. There is many formulation of the modularity, but we give the one found in [7]. The modularity of a partition $(C_1, \cdots, C_k)$ is defined by

$$\mathcal{M}(C_1, \cdots, C_k) = \sum_{i=1}^{k} \varepsilon_{i,i} - \sum_{i \neq j} \varepsilon_{i,j} \tag{6.23}$$

with

$$\varepsilon_{i,j} = \sum_{v \in C_i, u \in C_j} \omega(v, u) \tag{6.24}$$

where each edge is included at most once in the computation.

Let us remark that the modularity is directly linked to internal and external degrees (cf. expressions (6.14) and (6.13)):

$$deg_i(C_i) = \varepsilon_{i,i} \;\&\; deg_e(C_i) = \sum_{i \neq j} \varepsilon_{i,j}. \tag{6.25}$$

The quickest algorithms can compute the edge betwneenness on a graph in $O(n \cdot m)$ for unweighted graphs and in $O(n \cdot m + n^2 logn)$ for the weighted version [32], using for each vertice a single source shortest path problem using a modified version of Dijkstra's solution or a breadth-first search.

Figure 6.4 gives an illustration of communities found using the edge betweenness.

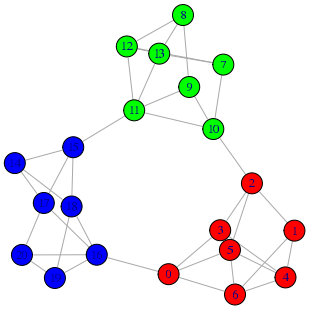


Fig. 6.4. A simple example of clustering based on betweenness

Cuts

Several community detection's algorithms are based on the minimization of the number of edges which link the clusters. These algorithms are, more or less, based on the minimization for each cluster of the cut size between the cluster and the outside of this latter. The *Kernighan-Lin algorithm* is one the first algorithm [33] in this way. These authors worked on the problem of partitioning electronic circuits onto boards, and the need of minimizing the number of connections between the cards. The motivation comes from the fact that there is a limit to the number of components on a card and, connections between cards have an high cost. They use, as quality criterion for the partition, the difference between the internal and the external degree of a cluster:

$$Q(C) = deg_i(C) - deg_e(C) \tag{6.26}$$

and they try to find the bi-partition of the graph which maximizes Q. To find the best subdivision in two clusters, they start with an arbitrary subdivision, compute Q, and then, subsets consisting of equal numbers of vertices are swapped between the two clusters to find when Q increases. The time execution is in $O(n^2 \log n)$. If we want to find k clusters, the procedure is repeated $k - 1$ times on the whole set.

Rather than using directly the cut size, another popular criterion for hierarchical divisive clustering on graph is the conductance (6.12), because it take into account the order of the sets that are being cut apart [7].

The two main problems with these divisive algorithms, is firstly the fact that find the bisection of the graph with the minimal cut size is an NP-complete problem [34] and find a cut wit the minimal conductance is NP-hard [35]. The second problem is shared by most hierarchical divisive algorithms: when to stop splitting the graph? One can fix the number of clusters at the beginning of the process, but other approaches exist like in [25], where authors propose to use the edge connectivity and the notion of highly connected subgraph (HCS) (see (6.16)): they do not split a subgraph which is also an HCS.

6.7 Spectral Methods

Spectral algorithms constitute a very particular class of techniques. And this particularity is to perform the classification based on eigenvectors of matrix build upon the adjacency (or weight) matrix.

In this part of the document, we suppose that G is an undirected, weighted graph, with positive symmetric weights matrix W (cf. expression (6.2)): $w_{i,j} = w_{j,i} \geq 0$. Moreover we need to define the *degree matrix* D:

$$D = W \cdot \mathcal{I} \tag{6.27}$$

where $\mathcal{I}$ is the identity matrix. D is such that we found the degrees $deg(v_i)$ on the diagonal. Now, we are able to define the spectral clustering: the *Laplacian matrix*:

$$L = D - W. \tag{6.28}$$

The more interesting property of L is directly linked to the notion of connected component, which is a connected subgraph A without connection with $V \setminus A$. Let us suppose that there is exactly k connected components in G, which form a partition of $G : C_1, \cdots, C_k$, with $c_i = |C_i|$. Without loss of generality, we can assume that the vertices can be ordered according to their connected component. Then, the adjacency matrix W has a block diagonal form, and L too:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix} \tag{6.29}$$

where each of the blocks L_i is a square matrix, of order c_i, which is the Laplacian of the connected component C_i. As for all block diagonal matrix, we know that the spectrum of L, e.g. the list of its eigenvalues denoted $\sigma(L)$, is equal to the union of the spectrum of the L_j. Moreover it can be shown [36] that the smallest eigenvalue of L is 0 and its multiplicity is exactly equal to

k, the number of connected components. Let us recall the classical similarity relation $L = Q\Lambda Q^{-1}$. And as L is a block diagonal matrix, Q and Λ too.

Now, let us suppose, that in each spectrum matrix Λ_i, eigenvalues are sorted in the decreasing order: $\lambda_{i_1} \geq \lambda_{i_2} \geq \cdots \geq \lambda_{i_{c_i}}$, where the last eigenvalue $\lambda_{i_{c_i}}$ is 0. Using expression (6.27) it is easy to show that the constant one vector $\mathbf{1}_{c_i} = (1, \cdots, 1)^t$ of dimension c_i is an eigenvector of this last eigenvalue:

$$L_i \cdot \mathbf{1}_{c_i} = D_i \cdot \mathbf{1}_{c_i} - W_i \cdot \mathbf{1}_{c_i} = \begin{pmatrix} d_1 \\ \vdots \\ d_{c_i} \end{pmatrix} - \begin{pmatrix} \sum \omega_{1,j} \\ \vdots \\ \sum \omega_{c_i,j} \end{pmatrix} = 0 \cdot \mathbf{1}_{c_i}. \tag{6.30}$$

As a direct consequence of this, the last vector column of Q_i is the constant one vector $\mathbf{1}_{c_i}$, and then the corresponding column in Q has the following form:

$$(\underbrace{0, \cdots, 0}_{\sum_{j=1}^{i-1} c_j}, \underbrace{1, \cdots, 1}_{c_i}, \underbrace{0, \cdots, 0}_{\sum_{j=i+1}^{k} c_j})^t. \tag{6.31}$$

These last column vectors of each block of Q are indicator vectors of the k connected components: there is a one only on the rows corresponding to indices of the nodes of the concerned connected component. Then if we denote $\mathbf{u_1} \cdots, \mathbf{u_k}$ these indicator vectors and build $U \in \mathbb{R}^{n \times k}$ with these latter, and denote $Y = U^t$ the transposition of U, for which columns vectors y_i belong to $\mathbb{R}^k$, then y_i contains only zeros except a one for the dimension corresponding to its connected component, i.e. if $v_i \in C_k$, then $y_{j,i} = 1$ if $j = k$, and $y_{j,i} = 0$ otherwise.

But define a community as a connected component is rather strict, and the introduction of one edges between two connected components causes some eigenvalues that were zero become slightly larger than zero but the underlying structure can be seen using the eigenvectors of the Laplacian and easily retrieved using a simple k-means on Y. In the general case, it remains to know k, but it is a problem shared with almost all clustering algorithms. Then, given a Laplacian L and a number of cluster k, the general algorithm for spectral clustering is the following:

1. compute the eigenvalues and sort them such $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$,
2. compute the last k eigenvectors $\mathbf{u_{n-k}} \cdots, \mathbf{u_n}$,
3. form matrix $U \in \mathbb{R}^{n \times k}$ with $\mathbf{u_{n-k}} \cdots, \mathbf{u_n}$ as columns, and matrix $Y = U^t$,
4. cluster the pointsy_i using the k-means into clusters $A_1, \cdots, A_k$,
5. build the communities $C_1, \cdots, C_k$ such $C_i = \{v_j | y_j \in A_i\}$.

For efficiency reasons it is preferable to use a normalized version of the Laplacian.

Shi and Malik [37] propose to use the following normalization

$$L_{\text{rw}} = D^{-1}L = I - D^{-1}W. \tag{6.32}$$

The notation L_{rw} is due to the fact that it is closely related to random walk. Ng, Jordan and Weiss [38] propose to use the following normalization

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \tag{6.33}$$

and furthermore impose to normalize U to have a norm equal to one. The notation L_{sym} is used because it is a symmetric matrix (see [39] for a complete review of the properties of the both normalizations). In practice it is preferable to use $L_{\text{rw}} = D^{-1}L$ because its eigenvectors are directly the cluster indicators $\mathbf{1}_{C_i}$, while the eigenvectors of L_{sym} are additionally multiplied by $D^{1/2}$, which might leads to undesirable effects. And moreover using L_{sym} also does not have any computational advantage [36].

Let us end on the complexity issue: spectral clustering requires the computation of the k eigenvectors of the Laplacian matrix, and if the graph is large, an exact computation require a time $O(n^3)$, which is penalizing for large graphs.

6.8 Galois Lattices

All the previous clustering methods are conceived to build non overlapping clusters, and are non exhaustive (i.e. one execution gives one solution), but there exists more informative methods. One of these interesting methods, is the Formal Concept Analysis (FCA) and the Galois Lattices [40] [41], which can be used for the conceptual clustering [42] [43]. A Galois Lattice clusters the data (called objects) in classes (called concepts) using their shared properties. The concepts found using Galois Lattices can be communities of people sharing connections, but also shared informations or opinions in an online network.

We introduce these lattices with a classical example, let us suppose that we have a description of some animal species (see table 6.1) : Lion, Finch, Eagle, Hare and Ostrich. This description is based of a list of properties such these species have or not: Ddo hey preying? Do they flying? Are they birds? Are they mammals? It is easy to compute a matrix of similarity counting the number of shared properties. As explained in Section 6.4 we can turn these similarities in distances and then apply an ascendant hierarchical clustering.

Table 6.1. A very simple incidence table description for a few animal species

	Preying	Flying	Bird	Mammal
Lion	x			x
Finch		x	x	
Eagle	x	x	x	
Hare				x
Ostrich			x	
Bee		x		
Bat		x		x

But in this case for each clustering one and only "point of view" will be shown. Figure 6.5 show a "multi-point of view" representation of the same informations, where species are labeled in red, while properties are labeled in blue. In this graph we see that Lions are the only ones (in our table) to be mammal and to prey, while Bats are the only ones to be mammal and to fly. In the same way, we see that Eagles are the only birds which both fly and prey, while it shares the two first properties with Finches. And these two flying birds, share this only property with Ostriches. We see in the graph that the concept "birds" is not the same that the concept "flying", because in the first one we have the "non-flying" Ostriches, and in the second one the "non-birds" Bats, which are mammals.

The reading rule of this kind of graph is the following: there is a downward path from a node v_i with a property p to a node v_j, only if v_j has also the property p. A consequence is the following: if there is two downward different paths from a node v_i with a property p to two different nodes v_j and v_k, then v_j and v_k share this property p.

This kind of graph is called a *Galois lattice*. The first use of lattices for observed structure data back to the attempt of formalize the logic of Quantum mechanics in [44]

We give now the formal definition of Galois lattices. Let us suppose that we have a set of objects O, and a set of possible attributes A for these objects. The possession of a property $a \in A$ by an object $o \in O$ is fulfilled when there is a relation I between them: aIo. Relations I between O and A are captured

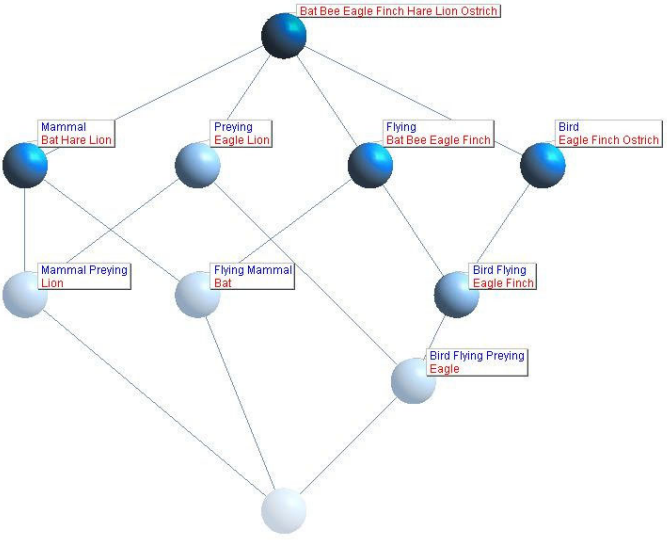


Fig. 6.5. Species Galois Lattice

in a binary incidence matrix. The triplet $K = (O, A, I)$ is called a *formal context* or simply a context.

A *concept* is any couple $C = (X, Y) \subset O \times A$, which can forms a closed rectangle in the incidence matrix, i.e. we must have:

$$f(X) = \{a \in A | \forall o \in X, oIa\} = Y \tag{6.34}$$

$$g(Y) = \{o \in O | \forall a \in Y, oIa\} = X. \tag{6.35}$$

Functions f and g defined in (6.34) and (6.35) are called respectively the *intension* and the *extension* of the concept C. The intention is the set of the defining properties of the concept, while the extension is the set of the objects which forms the concept. The couple (f, g) is called a *Galois connection*.

For the sake of illustration, in the table 6.1 the set $X = \{Finch, Eagle\}$ gives a concept, because $f(X) = \{Flying, Bird\} = Y$ and $g(Y) = X$, and this concept is $(\{Finch, Eagle\}, \{Flying, Bird\})$. On the contrary$X' = \{Lion, Hare\}$ do not gives a concept because $f(X') = \{Mammal\} = Y'$ and $g(Y') = \{Lion, Hare, Bat\}$, but this latter set gives the concept $(\{Lion, Hare, Bat\}, \{Mammal\})$.

Finally the *Galois lattice* is the set of concepts L with the following partial order $\leq$:

$$(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2 (\text{ or } Y_1 \supseteq Y_2). \tag{6.36}$$

The Galois lattice is denoted $T = (L, \leq)$ and its representation, as in figure 6.5, is called its *Hasse diagram*. As seen in the above example find a concept is not difficult:

1. pick a set of obects X,
2. compute $Y = f(X)$,
3. compute $X' = g(Y)$,
4. (X', Y) is a *formal concept* .

The dual approach can be taken starting with a set of attributes.

A more difficult task is, given a formal context table, to generate all the existing concepts and build the Hasse diagram of the Galois lattice. We detail here one of the simplest methods to build a Galois lattice, the Bordat's algorithm [45], which build L and its Hasse diagram.

Let us, firstly, define the cover of a concept $C = (X, Y)$, denoted $\underline{C}$

$$\underline{C} = \{C' | C' \leq C \text{ and } \exists C'' : C' \leq C'' \leq C\}. \tag{6.37}$$

The algorithm starts with the set of concept given by the single objects $(o, f(o))$ and then find all their children nodes which are added to L and linked to their parent. This child generation process is iteratively repeated for every new pair:

- $L < -\{(o, f(o)) | o \in O)\}$,
- for each concept $C \in L$:

  - build $\underline{C}$,
  - for each $C' \in \underline{C}$:
    - if $C' \notin L$, then $L < -L \cup \{C'\}$,
    - add the edge between C and C'.
  - end for.
- end for.

Several other algorithms were proposed and the reader can take a look at [46] for a review and a comparison of performances.

The first use of Galois lattices in social network data analysis is owed to Freeman in [47], where he use the belonging to a clique as attribute. Another direct use of Galois lattices in the social network analysis is to use the adjacency matrix as incidence matrix. In the figure 6.6 we give an simple

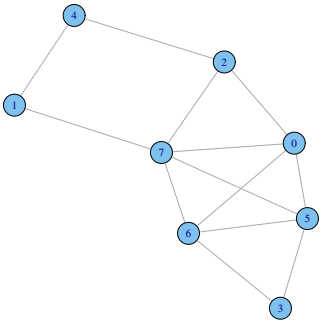


Fig. 6.6. A simple graph example

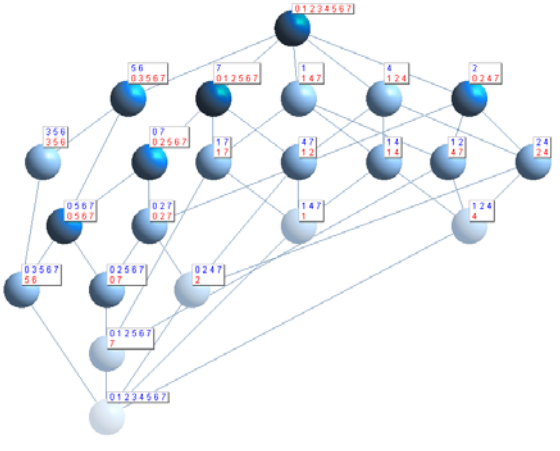


Fig. 6.7. The Galois Lattice of Social Network shown in the figure 6.6

example of a graph containing three maximal cliques : $\{3, 5, 6\}$, $\{0, 2, 7\}$ and $\{0, 5, 6, 7\}$. In figure 6.7 we give the Galois lattice of this network, and even if it is look more complicated than the network, it is very easy to retrieve the cliques : it suffice to find concepts where both objects and attributes sets are equal. We surely could find concept like k-plex or k-core in the same way.

Now, let us take a look to the complexity issue. If we denote $|O|$ the number of objects, $|A|$ the number of attributes and $|L|$ the size of the lattices (i.e. the number of concepts), then the different algorithms have a complexity time in $O(|O|^2|A||L|)$ or $O(|O||A|^2|L|)$ (see [46] for details).

The Galois lattice based clustering is then costly, and moreover not simple to read but , but has several advantages in comparison with similarity clustering (see [48] for a good comparison). First, the notion of similarity is more strict in Galois lattices than in similarity based clustering: two objects in a Galois lattice are similar if they share identical properties, while they are alike to a degree quantified by a proximity measure in the other case. Secondly, given a dataset the Galois lattice turns a context into a unique and complete lattice of concepts, while classical clustering produces a result over all possible classes, depending of several choices (methods, parameters,...). Moreover, the resulting classes are, in most of the methods, disjoint whereas concepts in the lattice may overlap.

6.9 Discussion

As already pointed out forward the essential starting point in community detection, the definition of a community or cluster, is not an easy task [13]. Already in 1964, Bonner [49] argued that there could not be a universal definition of a cluster. In fact, what is a cluster is in the eyes of the beholder, and *one person's noise could be another person's signal* [3]. That is why we can write that *cluster analysis is structure seeking although its operation is structure imposing* [50].

Even with a given single and clear definition of a cluster, we expect from an adapted clustering that it gives a good clustering, but, again, what is a good clustering? Knowing that clustering algorithms transform clusters's research into optimization problem whose computational complexity is typically intractable and which are solved by approximation algorithms [13]. And in data clustering many choices must be done before any analysis (cluster definition , algorithms, measures, ...) which influence strongly the result. And the crucial choice of the algorithm will be influenced by the sizes (number of vertices and/or number of edges) of the network, which has a direct impact on the time of execution.

But, in spite of all these warnings, clustering algorithms allow us to retrieve valuable pieces of information in social networks, by finding communities. But in the future, the community detection is not the only classification task in social networks analysis.

References

1. Freeman, L.: The Development of Social Network Analysis: A Study in the Sociology of Science. Empirical Press, Vancouver (2004)
2. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge Univ. Press (1994)
3. Han, J., Kamber, M.: Data mining: concepts and techniques. Morgan Kaufmann (2006)
4. Hand, D.J., Mannila, H., Smyth, P.: Principles of data mining. MIT Press (2001)
5. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69(2), 026113 (2004)
6. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 75–174 (2010)
7. Shaeffer, S.E.: Graph clustering. Computer Science Review 1, 27–64 (2007)
8. Moreno, J.L.: Emotions mapped by new geography (1933)
9. Cartwright, D., Harary, F.: Structural balance: A generalization of heider's theory. Psychological Review 63, 277–293 (1956)
10. Scott, J.: Social Network Analysis. Sage (2000)
11. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33, 452–473 (1977)
12. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99, 7821–7826 (2002)
13. Estivill-Castro, V.: Why so many clustering algorithms: a position paper. SIGKDD Explor. Newsl. 4(1), 65–75 (2002)
14. Luce, R.D., Perry, A.D.: A method of matrix analysis of group structure. Psychometrika 14(2), 95–116 (1949)
15. Luce, R.D.: Connectivity and generalized cliques in sociometric group structure. Psychometrika 15, 169–190 (1950)
16. Mokken, R.: Cliques, clubs and clans. Quantity and Quality 13, 161–173 (1979)
17. Seidman, S., Foster, B.: A graph-theoretic generalization of the clique concept. Journal of Mathematical Sociology 6, 139–154 (1978)
18. Seidman, S.B.: Internal cohesion of ls sets in graphs. Social Networks 5(2), 97–107 (1983)
19. Luccio, F., Sami, M.: On the decomposition of networks in minimally interconnected subnetworks. IEEE Transaction Circuit Theory 16(2), 184–188 (1969)
20. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proc. Natl. Acad. Sci. USA 101, 2658–2663 (2004)
21. Bollobas, B.: Modern Graph Theory. Springer, New York (1998)
22. Kannan, R., Vempala, S., Vetta, A.: On clusterings - good, bad and spectral. Journal of the ACM 51(3), 497–515 (2004)
23. Chung, F.R.K., Lu, L.: Complex graphs and networks. AMS (2006)
24. Freeman, L.: A set of measures of centrality based upon betweenness. Sociometry 40(1), 35–41 (1977)
25. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Inf. Process. Lett. 76(4-6), 175–181 (2000)

26. Burt, R.S.: Positions in networks. Social Forces 6, 93–122 (1976)
27. Lorrain, F., White, H.: Structural equivalence of individuals in social networks. Journal of Mathematical Sociology 1, 49–80 (1971)
28. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif., vol. 1, pp. 281–297 (1967)
29. Diday, E., et al.: Optimisation en classification automatique. vol. i, ii. Technical report, Institut National de Recherche en Informatique et en Automatique (INRIA), Le Chesnay, France (1979)
30. Lance, G., Williams, W.: A general theory of classificatory sorting strategies. i: Hierarchical systems. Comp. Journal 9, 373–380 (1967)
31. Newman, M.E.J.: Detecting community structure in networks. The European Physical Journal B 38, 321–330 (2004)
32. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25, 163–177 (2001)
33. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49(2), 291–308 (1970)
34. Bui, T.N., Leighton, F.T., Chaudhuri, S., Sipser, M.: Graph bisection algorithms with good average case behavior. Combinatorica 7, 171–191 (1987)
35. Sima, J., Schaeffer, S.: On the NP-Completeness of Some Graph Cluster Measures. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 530–537. Springer, Heidelberg (2006)
36. von Luxburg, U.: A tutorial on spectral clustering. Technical Report Technical Report 149, Max Planck Institute for Biological Cybernetics (2006)
37. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
38. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14. MIT Press (2002)
39. Chung, F.R.K.: Spectral Graph Theory. CBMS Regional Conference Series in Mathematics, vol. (92). American Mathematical Society (1997)
40. Barbut, M., Monjardet, B.: Ordre et classification, Algebre et combinatoire, Tome 2. Hachette (1970)
41. Birkhoff, G.: Lattice Theory, vol. 25. American Mathematical Society, New York (1940)
42. Carpineto, C., Romano, G.: Galois: An order-theoretic approach to conceptual clustering. In: Proc. Of the 10th Conference on Machine Learning, pp. 33–40. Morgan Kaufmann, Amherst (1993)
43. Wille, R.: Line diagrams of hierarchical concept systems. Int. Classif. 11, 77–86 (1984)
44. Birkhoff, G., von Neumann, J.: The logic of quantum mechanics. Ann. Math. 37, 823–843 (1936)
45. Bordat, J.: Calcul pratique du treillis de Galois d'une correspondance. Mathématique, Informatique et Sciences Humaines 24(94), 31–47 (1986)
46. Kuznetsov, S.O., Obedkov, S.A.: Comparing performance of algorithms for generating concept lattices. In: Concept Lattices-based Theory, Methods and Tools for Knowledge Discovery in Databases (CLKDD 2001), Stanford, July 30 (2001)

47. Freeman, L.: Cliques, Galois lattices, and the structure of human social groups. Social Networks 18, 173–187 (1996)
48. Valtchev, P., Missaoui, R.: Similarity-based clustering versus Galois lattice building: Strengths and weaknesses. In: Workshop Objects and Classification, A Natural Convergence European Conference on Object-Oriented Programming (2000)
49. Bonner, R.: On some clustering techniques. IBM Journal of Research and Development 8, 22–32 (1964)
50. Aldenderfer, M.S., Blashfield, R.K.: Cluster Analysis. Sage Publication (1984)

7

Semantic Technologies and Triplestores for Business Intelligence

Marin Dimitrov

Ontotext AD
Sofia, Bulgaria
marin.dimitrov@ontotext.com

Summary. The Semantic Web is the next generation Web of data, which extends the current Web with means to provide well-defined meaning of information and easily find, integrate and analyze relevant information from different sources. Semantic Databases, or triplestores, play a very important role in the realization of the Semantic Web vision, since they provide the means to integrate, store and query the vast amounts of metadata generated on the Semantic Web every day. This paper presents a brief overview of the Semantic Web goals and related standards, with a special focus on the advantages, design and performance factors for triplestores. Finally, the paper provides an overview of Business Intelligence related scenarios where Semantic Technologies and triplestores in particular provide valuable advantage and differentiation.

Keywords: semantic web, RDF, data management, business intelligence.

7.1 Introduction

The Semantic Web is the next generation Web of data, which extends the current Web with means to provide well-defined meaning of information and easily find, integrate and analyze relevant information from different sources.

The goal of this paper is to provide a brief summary of the most prominent Semantic Web related standards and technologies (section 7.2) with particular focus on the Resource Description Framework (RDF), Web Ontology Language (OWL), SPARQL query language and Linked Data.

Section 7.3 provides an overview of RDF databases (also called triplestores), the specific advantages over other traditional data management systems, design issues and performance factors, overview of current commercial and open source triplestores, as well as the current approaches to benchmarking the performance of such systems.

Finally, section 7.4 outlines some Business Intelligence related scenarios where using Semantic Technologies and triplestores can provide valuable advantage and differentiation.

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 139 – 155, 2012.

7.2 Semantic Technologies

This section provides a summary of the vision of the Semantic Web, the benefits of applying Semantic Technologies and a brief overview of the current Semantic Web related standards.

7.2.1 The Vision of the Semantic Web

The Semantic Web is the next generation Web of data, which – as envisioned by Tim Berners Lee a decade ago [1] – will become *"an extension of the current web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation."*

The goals of the Semantic Web can be summarized as:

- Extend the principles of the Web from documents to data
- Data should be accessed using the general Web architecture and protocols
- Data should be related to one another just as documents are already
- Provide a common framework that allows data to be shared and reused across applications, data to be processed automatically and new relationships between pieces of data to be automatically inferred

Semantic Technologies are already being adopted by enterprises in various domains such as Enterprise Data Integration, Life Sciences, intelligent search, cultural heritage, content discovery and recommendation. An extensive list of case studies and use cases of applying Semantic Technologies within various enterprises is provided in [2].

7.2.2 Semantic Web Timeline

Since 1999 W3C has provided a rich family of Semantic Web related standards for modeling knowledge and rules, querying of knowledge bases, embedding semantic annotations in web pages and web service component descriptions, mapping relational data to RDF and providing provenance information for data on the Semantic Web. This section provides a brief summary of the timeline of the most prominent Semantic Web related standards (Fig. 7.1): Resource Description Framework (RDF), Web Ontology Language (OWL), the SPARQL query language, Rule Interchange Format (RIF), RDFa annotations for web documents, semantic annotations for Web Services (SAWSDL), Simple Knowledge Organization Schema (SKOS), approaches for mapping relational data to RDF, and finally the Provenance Interchange Language (PIL).

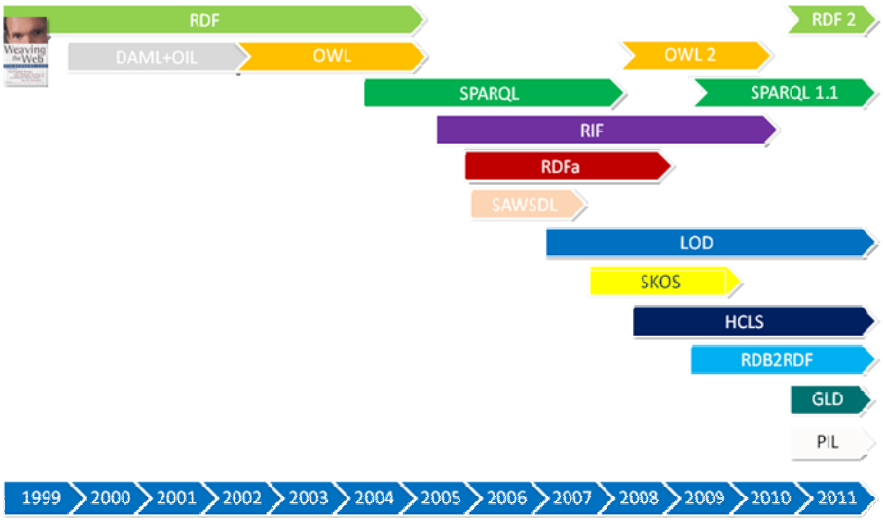


Fig. 7.1. Semantic Web Timeline, 1999-2011

Resource Description Framework (RDF). RDF [3] is a W3C standard providing a data model for the Semantic Web. Resources on the Semantic Web are described in terms of statements, also called *triples*, composed of a *subject*, *predicate* and an *object*. The subject and the object of a triple denote things (or entities) in some domain, and the predicate represents the relationship that holds between the subject and the object. The subject and the object can be represented as nodes in a graph, while the predicate is a directed arc connecting the two nodes. Several triples can be connected together into a bigger RDF graph. More details on RDF are available in section 7.2.3.

Web Ontology Language (OWL). OWL is a W3C standard [4][5] defining a family of ontology representation languages with different levels of expressivity. There are three sublanguages defined:

- *OWL Lite*, with limited expressivity but logical decidability and low formal complexity;
- *OWL DL,* which provides maximum expressivity while retaining completeness and decidability and is suitable for efficient reasoning based on Description Logics;
- *OWL Full,* which provides maximum expressiveness but provides no computational guarantees and is not suitable for efficient reasoning algorithms.

Additionally, the OWL 2 version of the standard defines three profiles: *OWL 2 EL* which has polynomial time reasoning complexity and is suitable for application scenarios with very large ontologies; *OWL 2 QL* which is designed for

very efficient query answering over large volumes of instance data; and *OWL 2 RL* which provides a balance between expressivity and scalable reasoning and is suitable for rule-based reasoning algorithms.

More details on OWL are available in section 7.2.3.

SPARQL Protocol and RDF Query Language (SPARQL). SPARQL [6] is the W3C standardized language for querying over RDF data. A SPARQL query is comprised of triple patterns, conjunctions, disjunctions and optional patterns.

Rule Interchange Format (RIF). RIF is W3C standard for exchanging rules between rule systems [7]. RIF defines a family or languages (also called dialects), with different expressivity and complexity.

RDF in Attributes (RDFa). RDFa [8] is a W3C standard extension to XHTML that makes it possible to embed RDF metadata into web documents. As a result, web content authors can easily augment their content with interoperable machine readable data that can be utilized by intelligent agents and applications.

Semantic Annotations for WSDL (SAWSDL). SAWSDL [9] is a W3C standard that defines extension attributes to the Web Services Description Language (WSDL) that allows semantic annotations and references to ontologies to be embedded within WSDL component descriptions of Web Services.

Simple Knowledge Organization Schema (SKOS). SKOS [10] is W3C standard that provides a lightweight formal language and data model for thesauri, taxonomies and classification schemes. SKOS is based on RDF and its main goal is to provide means for easy and low-cost migration of legacy knowledge organization systems to the Semantic Web.

RDB2RDF. RDB2RDF is a set of W3C recommendations (in progress) that will standardize a language for mapping relational data and relational schemata to RDF and OWL. The latest working draft of the R2RML mapping language is available at [11], while the latest working draft of the mapping from relational to RDF data is available at [12].

Provenance Interchange Language (PIL). PIL [13] is a work in progress at W3C, aiming to provide a standard language for exchanging provenance information across applications. The design goals of PIL, according to the W3C, are: to be applicable not only to Semantic Web objects, but to any kind of web resource; to have a low entry point; to have a small core model which is easily extensible.

7.2.3 RDF, OWL and SPARQL

RDF provides a simple data model that makes it possible to formally describe the semantics of information in a machine accessible way. RDF forms the basis of the

Semantic Web and other related standards are built on top of it. Entities, or things on the Semantic Web are called *resources*, and RDF provides a simple way to make *statements* (also called *triples*) about such resources.

A triple is comprised of a *subject*, which is a resource with a globally unique identifier (URI), an *object*, which can be another resource or a literal value, and a *predicate*, which is a named relation that holds between the subject and the object. The predicate also has its own a globally unique identifier (URI). The subject and the object can be represented as nodes in a graph, while the predicate is the directed arc connecting the two nodes. Several triples can be connected together into a bigger RDF graph.

The Subject-Predicate-Object model of RDF is very similar to classic data modeling approaches such as the Entity-Attribute-Value (EAV) model or Entity-Relationship Model (ERM).

There are several formats in which RDF data can be published and interchanged on the Semantic Web, including N-Triples [14], N3 [15], and RDF/XML [16].

The RDF data model itself is very simple and creates graph-like structures but without any notion of classes, properties or inheritance, which are essential for modeling of ontologies and taxonomies. RDF Schema (RDFS) is a schema language which extends the generic RDF model with means for defining classes (*rdf:type* and *rdfs:Class*), properties (*rdfs:Property*), domain/range relations between the subjects and the objects in a triple (*rdfs:domain/rdfs:range*), and hierarchies of classes and properties (*rdfs:subClassOf* and *rdfs:subPropertyOf*). RDFS also defines a set of logical entailment rules (Fig. 7.2), which make it possible for new, implicit triples to be inferred by the explicit (asserted) triples in an RDF database.

1:	$s\ p\ o$ (if o is a literal)		$\Rightarrow$ _:n rdf:type rdfs:Literal
2:	p rdfs:domain x	& $s\ p\ o$	$\Rightarrow$ s rdf:type x
3:	p rdfs:range x	& $s\ p\ o$	$\Rightarrow$ o rdf:type x
4a:	$s\ p\ o$		$\Rightarrow$ s rdf:type rdfs:Resource
4b:	$s\ p\ o$		$\Rightarrow$ o rdf:type rdfs:Resource
5:	p rdfs:subPropertyOf q	& q rdfs:subPropertyOf r	$\Rightarrow$ p rdfs:subPropertyOf r
6:	p rdf:type rdf:Property		$\Rightarrow$ p rdfs:subPropertyOf p
7:	$s\ p\ o$	& p rdfs:subPropertyOf q	$\Rightarrow$ $s\ q\ o$
8:	s rdf:type rdfs:Class		$\Rightarrow$ s rdfs:subClassOf rdfs:Resource
9:	s rdf:type x	& x rdfs:subClassOf y	$\Rightarrow$ s rdf:type y
10:	s rdf:type rdfs:Class		$\Rightarrow$ s rdfs:subClassOf s
11:	x rdfs:subClassOf y	& y rdfs:subClassof z	$\Rightarrow$ x rdfs:subClassOf z
12:	p rdf:type rdfs:ContainerMembershipProperty		$\Rightarrow$ p rdfs:subPropertyOf rdfs:member
13:	o rdf:type rdfs:Datatype		$\Rightarrow$ o rdfs:subClassOf rdfs:Literal

Fig. 7.2. RDFS entailment rules

The following short example illustrates how the RDFS entailment rules work. In this example we define three classes (*mammal*, *human*, *man*) and one individual (*John*) as follows:

```
:human rdfs:subClassOf :mammal .
:man rdfs:subClassOf :human .
:John a :man .
```

When the triplestore applies the RDFS entailment rules (Fig. 7.2) to the set of explicit statements above, the following new statements will be derived and added to the knowledge base:

```
:man rdfs:subClassOf :mammal . (from Rule 11)
:John a :human . (from Rule 9)
:John a :mammal . (from Rule 9)
```

OWL additionally extends RDF and RDFS and provides more expressive modeling constructs that make it possible to build very complex ontological models and knowledge organization schemata. In particular, OWL provides powerful modeling constructs such as:

- Complex class expressions – intersection, union and complement of classes
- Equivalence / disjointness of classes
- Identity between resources (*owl:sameAs*)
- Object / Datatype properties
- Cardinality restrictions
- Transitive, symmetric, inverse, functional and reflexive properties
- Annotations

As already mentioned in section 7.2.2, OWL defines three layered sublanguages (also called dialects) with different expressivity and logical complexity: OWL Lite, OWL DL and OWL Full, as well as three profiles, which are designed to provide maximum performance for specific application scenarios: OWL 2 EL, OWL 2 QL and OWL 2 RL.

SPARQL is the query language for RDF and OWL data. Unlike SQL, the SPARQL standard defines not only the query language syntax and semantics, but also a simple protocol for querying remote SPARQL endpoint (RDF databases) over HTTP. Additional advantages of SPARQL over traditional SQL include: query entities even when the relationships between them are unknown; query any combination of local and disparate RDF databases in a single query; transform data between vocabularies or schemata (similar to XSLT).

There are four types of SPARQL queries:

- *Select* – query data with complex graph patterns
- *Ask* – similar to a Select query, but instead of returning any data, this query will only return a boolean answer that indicates whether any data would be returned
- *Describe* – returns all triples about a particular resource
- *Construct* – constructs new triples based on the query resultset bindings

A SPARQL query has the following components:

- *Prefix shortcuts* (optional), which make it possible to refer to RDF resources with shorter identifiers, as opposed to the full URIs
- *Query result clause*, which defines the variable bindings that will be returned in the resultset
- *Datasets* (optional), which refer to the RDF graphs or data sources that will be queries
- Arbitrary complex *triple patterns* and *filters* that will be matched over the RDF graph
- *Query modifiers* (optional), which specify the resultset ordering, cursors (offset/limit), filtering of duplicates, etc.

7.2.4 Linked Data

The principles of "linked data" were first introduced by Tim Berners-Lee [17]. A more detailed overview of Linked Data is available in [18]. In summary, Linked Data provides means for publishing, interlinking and consuming RDF datasets on the Web by adhering to four basic principles, also known as the *Linked Data Principles*:

1. *Use URIs as names for things.*
2. *Use HTTP URIs, so that people can look up those names.*
3. *When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).*
4. *Include links to other URIs, so that they can discover more things.*

The basic idea is to apply the design principles of the World Wide Web, but instead of providing a framework for publishing, interlinking and consuming of documents, to provide a framework for sharing structured datasets (usually, but not necessarily represented in RDF).

In a nutshell, the idea behind the 1st principle is to make it possible to identify "things" (such as resources or collections of resources) on the data web.

The 2nd principle recommends that only HTTP URIs are used for identifying Linked Data resources (as opposed to alternative URI schemes such as DOI [19] or URN [20]). The rationale behind this principle is to provide a simple, yet decentralized way to create globally unique identifiers for resources on the data web. Additionally, an HTTP URI makes it easy to access the particular resource with existing widely adopted tools.

The 3rd principle mandates that HTTP URIs are dereferenceable, i.e. HTTP clients can access the description of a resource on the data web by looking up the resource URI over the HTTP protocol.

Finally, the 4th principle recommends that resource descriptions on the data web link to other resources (or collection of resources) so that applications can discover and access additional data about some resource.

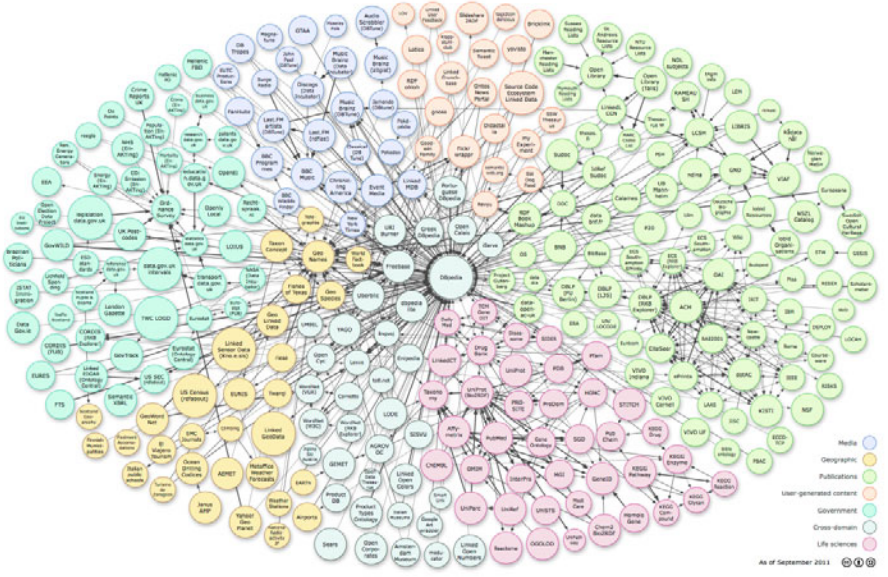


Fig. 7.3. Linking Open Data cloud diagram, by R. Cyganiak and A. Jentzsch, Sep 2011

As of September 2011, the Linked Open Data cloud [38] includes more than 300 datasets in various domains (general purpose, geographic, government, media, libraries, life sciences, etc.), comprised of more than 30 billion triples (Fig. 7.3).

7.3 Triplestores

This section provides mode details about triplestores, and in particular: comparison to traditional data management systems, design factors, performance factors and benchmarks, and a summary of the most popular commercial and open source triplestores.

7.3.1 Comparison to Traditional DBMS

Triplestores as a data management technology bear many similarities to traditional database management systems like OLTP / OLAP databases and also relatively new NoSQL based systems. At the same time, triplestores have many differentiating features, which make them very suitable for specific application scenarios. The following sections provide a brief comparison of the similarities and differences between triplestores and other data management systems like RDBMS, multi-dimensional databases and NoSQL databases.

RDBMS. Traditional OLTP databases are optimized for high-volume of transactional data updates, static schema and relatively simple queries. Triplestores can rarely match the performance and scalability provided by RDBMS in application scenarios with well structured data and static schema. On the other hand, triplestores provide certain advantage when the data stored is mostly unstructured or semi-structured, sparse and graph-based data with evolving or dynamic schema. Additionally, a triplestore can infer new, implicit data based on sound logical rules.

Multidimensional Databases. OLAP databases are optimized for complex queries over terabytes of static data. The query performance and volume of data handled by an OLAP database can rarely be matched by a triplestore in such application scenarios with static schema and well structured and static data. On the other hand, a triplestore is very suitable for complex graph-based queries over sparse, unstructured/semi-structured data with dynamic schema. Even though some materialization for the dimensions is provided within the OLAP database, a triplestore can infer more implicit data based on sound logical rules.

NoSQL Databases. NoSQL databases are optimized for high-volume data updates and thousands of concurrent but very simple queries. Similar to triplestores, NoSQL databases also provide a relatively dynamic data model, suitable for storing semi-structured and sparse data, but the very limited expressivity of the NoSQL query languages, the lack of standard logical inference of implicit data, and the lack of standards may be a drawback in certain application scenarios.

Graph Databases. Graph databases are a subset of NoSQL databases, which bear many similarities to triplestores, in particular: dynamic data model, ability to efficiently store and query unstructured and semi-structured sparse data, expressive graph-based queries. At the same time, triplestores provide the advantage of a standard data model, a standard and more expressive query language; a standard data interchange format and logical inference of implicit data.

7.3.2 Triplestore Advantages

As already outlined in the previous section, triplestores provide specific differentiation and advantages over traditional data management approaches in many application scenarios.

The most significant differentiating features of triplestores can be summarized as (not in any particular order of importance):

1. *Globally unique identifiers* of entities in the RDF data model. As already described in section 7.2.3, resources (or entities) in RDF are identified by URIs, which lowers the cost of incremental data integration from disparate data sources and databases.

2. *Inference of implicit facts*. Triplestores provide means to infer new knowledge, based on the sound logical entailment rules of RDF and the various OWL sublanguages and dialects.
3. *Graph data model,* which is very suitable for unstructured, semi-structured and sparse data and additionally lowers the cost of incremental data integration.
4. *Agile schema,* which makes it easy for the schema and the instance data to evolve and adapt to changes in the application requirements. When a new relation (predicate/property) between entities is added on-the-fly to the RDF data model, no changes are required to existing instance data and the triplestore will infer new implicit data if necessary and make it immediately available for querying.
5. *Exploratory queries against unknown schema.* Query patterns can match data in the triplestore even when the entity types or the exact relationships between them are not known in advance, i.e. the syntax of the query does not need to exactly match the syntax of the data assertions in the triplestore.
6. *Compliance to standards*. RDF, OWL and SPARQL are well established and adopted standards which lowers the cost of interoperability, data integration and application integration. This is not the case with some database technologies like graph databases or NoSQL databases.

7.3.3 Design and Advanced Features

There are many design and implementation decisions that determine the functionality, performance and scalability, deployment requirements and long term TCO of a triplestore.

Storage Engine. There are several implementation approaches: from native triplestores, to triplestore which are built on top of a relational database, to triplestores built on top of a NoSQL engine.

Reasoning Strategy. Triplestores employ different strategies for the rule-based reasoning. Some perform *forward-chaining*, where the triplesore performs the entailment rules (Fig. 7.2) starring with the explicit statements and derives new valid implications. The process continues until the full inferred closure is computed and no new facts can be derived. An alternative strategy is *backward-chaining*, where at runtime the reasoner starts from a query and decomposes it into simpler requests or statements that can either be matched directly to known (explicit) statements, or further decomposed recursively into simpler requests or statements. A third alternative is the *hybrid strategy*, where partial materialization is performed at data loading time through forward-chaining, and part of the query patterns are matched through backward-chaining at run time.

Invalidation Strategy. Triplestores that employ forward-chaining based materialization need a specific strategy for *truth maintenance*, i.e. when a set of explicit statements is deleted from the repository, the implicit statements which are no longer inferable need to be deleted as well, since they are no longer logical consequences of the facts in the knowledge base. The overhead of keeping detailed meta-information about the inference dependencies is usually too high to be practically feasible. Different triplestores employ different strategies for truth maintenance: from the most simple and inefficient ones where the complete inferred closure needs to be recomputed even after a single deletion, to very advanced techniques which detect which part of the inferred closure is no longer valid and thus needs to be deleted as well.

***owl:sameAs* Optimization.** The *sameAs* predicate in OWL provides means for declaring that two different URIs denote the very same resource. The practical use for it is to align URIs that are used in different databases or data sources for the same real-world entity (e.g. *fbase:guid.9202a8c04000641f8000000000028aa7, dbpedia:Montreal* and *geonames:6077244* are the identifiers from Freebase, DBpedia and GeoNames respectively, which denote the resource describing the city of Montreal). Since *owl:sameAs* is a transitive, reflexive and symmetric relation, a lot of additional entailment rules will be activated and derive new implicit statements. As a result, *owl:sameAs* can significantly "inflate" the number of statements in the database and deteriorate the inference and query performance. Some triplestores employ specific optimizations, so that equivalent URIs are treated in a specific way that does not inflate the database indices, but can still be inferred when needed at run time.

7.3.4 Popular Triplestores

This section provides details of the most popular commercial and open source triplestores available on the market.

4store. 4store [21] is an *open source* triplestore developed by Garlik. 4store can run in a distributed cluster of up to 32 nodes. The data is fully partitioned across the cluster nodes (more details on the design & architecture are available in [22]). 4store itself does not provide any inferencing capabilities, though there is an external reasoner, 4s-reasoner [23], which provides backward-chaining based RDF reasoning on top of 4store.

AllegroGraph. AllegroGraph [24] is a commercial triplestore developed by Franz Inc., which supports SPARQL, RDFS+, and Prolog reasoning. Its features include: ACID transactions, online backups, text indexing, user-defined indices, compression, replication & standby cluster. AllegroGraph does not perform reasoning and materialization during the data load phase, but reasoning can be performed at runtime during query processing.

Bigdata. Bigdata [25] is an *open source* triplestore developed by SYSTAP. It provides a horizontally partitioned storage engine with RDF and limited OWL reasoning capabilities. The materialization itself is hybrid, with some entailments materialized at load time (forward chaining) and others at query time (backward chaining). Advanced features of Bigdata include full-text indexing and statement-level provenance.

Dydra. Dydra [26] is an RDF cloud service, i.e. a triplestore available as a Software-as-a-Service (SaaS), developed by Datagraph. It provides multi-tenant data store and a standard SPARQL endpoint on top of it. The triplestore operations (uploading, deleting and managing data) are performed through a REST API. The Dydra storage engine does not provide RDF reasoning and materialization capabilities at present.

Jena TDB. TDB [27] is an *open source* RDF storage engine for the Jena Semantic Web framework. TDB is non-transactional and provides a standard SPARQL endpoint. TDB relies on the Jena framework for RDF and limited OWL reasoning capabilities, but custom and user-defined rules are also supported.

Oracle. Oracle offers an RDF engine [28] as part of its Spatial offering since release 10g. The triplestore supports RDF and limited OWL inferencing and materialization. Inference itself is not performed at data load time, but only if initiated by the database administrator. Deletions of statements require a new full materialization. Advanced features include: data partitioning and compression for Real Applications Cluster (RAC), *owl:sameAs* optimization, fine-grained security, data versioning. RDF data can be queried with SPARQL or SQL.

OWLIM. OWLIM [29] is a commercial triplestore developed by Ontotext. It performs forward chaining based materialization and reasoning with respect to the semantics of RDF, OWL 2 RL, OWL 2 QL and a subset of OWL Lite / OWL DL. Additionally, user-defined rulesets are supported for optimal performance and expressiveness. Advanced features include: replication cluster, automated fail-over and load-balancing, *owl:sameAs* optimizations, full-text indexing & search within SPARQL, geo-spatial extensions, scalable RDF Rank, spreading activation based RDF Priming (that allows for the "priming" of large datasets with respect to concepts relevant to the context and to the query), scalable remote notifications.

Sesame. Sesame [30] is an *open source* semantic repository developed by Aduna. It provides a plugable storage and inference layer (SAIL) which makes it possible for different storage engines and reasoners to be plugged-in and used with the Sesame framework (several triplestores described in this chapter are compatible with the SAIL APIs). Sesame also provides a native storage engine and can be used without any 3rd party triplestore components.

StarDog. StarDog [31] is a commercial triplestore developed by Clark & Parsia LLC. StarDog provides backward-chaining based reasoning at query time, with respect to OWL 2 DL and all OWL 2 profiles (QL, EL and RL). Advanced features include full-text indexing and search, stored procedures, embedded machine learning functionality, compression.

Virtuoso. Virtuoso [32] is a commercial triplestore developed by OpenLink. There is also an *open source* community edition without some of the advanced features. Virtuoso itself is a "universal server" for data management not only of RDF but also of XML and relational data. By default Virtuoso performs backward-chaining at run time, though forward-chaining based materialization is also possible with SPARQL Update statements that generate the inferred triples.

7.3.5 Performance Factors

Measuring the performance of triplestores is a complex task that should consider various performance aspects and usage scenarios. In [33] a conceptual framework for benchmarking of triplestores is provided, which takes into consideration the main tasks performed by a triplestore and the various factors that affect its performance. The conceptual framework provides a detailed description of the *tasks* and *performance factors* for triplestore benchmarking, which we will briefly summarize.

Tasks. The typical tasks performed by a triplestore which need benchmarking, are identified in [33] as follows:

- *Data loading*, including storing and indexing new RDF data into the triplestore;
- *Query evaluation*, including the query evaluation, planning, optimization and data fetching;
- *Data modification,* including updates or deletions of instance data or the ontologies and schemata describing the data.

Performance Factors. Various factors have impact on the performance of the triplestore during the data loading task:

- *Materialization*, forward-chaining – if performed at data loading time – has significant impact on the performance of the data loading task;
- *Data model complexity*, the computational cost of using different logical models varies significantly, e.g. the complexity of working with RDFS models is much lower than the complexity of OWL 2 RL, which in turn is lower than the complexity of OWL DL;
- *Indexing specifics*, various indices can be created in order to optimize the query performance, and this will have an impact on the data loading time.

Several factors affect the performance of the query evaluation task:

- *Deduction*, whether backward-chaining is performed at query time to infer implicit triples;
- *Query complexity*, some RDF graph patterns and operators are more resource consuming and will result in reduced query performance;
- *Result-set size*, accessing and fetching large amounts of data will impact the query performance;
- *Number of concurrent clients* will also affect the query performance of the triplestore.

7.3.6 Benchmarks for Triplestores

There exist several popular benchmarks which measure the performance of triplestores. The benchmarks provide pre-defined test datasets of various sizes as well as pre-defined queries of different complexity. The following sections provide more details about the LUBM, BSBM and the SP^{2}Bench benchmarks for triplestores and SPARQL engines.

LUBM. The Lehigh University Benchmark (LUBM, [34]) provides ways to benchmark the storing and querying performance of triplestores on large datasets. There are 14 pre-defined queries and a data generator tool that can generate test datasets of various size and complexity.

BSBM. The Berlin SPARQL Benchmark (BSBM, [35][36]) provides a benchmark suite for measuring the querying performance of triplestores. The benchmark defines three use cases: Explore (12 distinct SPARQL queries), Explore & Update (17 distinct SPARQL queries) and Business Intelligence (8 distinct SPARQL queries).

SP^{2}Bench. The SPARQL Performance Benchmark (SP^{2}Bench, [37]) provides a benchmark specifically designed for testing of the most common SPARQL constructs, operators and RDF access patterns. It defines 12 SPARQL queries which have been designed to be application-agnostic and to cover the most common performance challenges for SPARQL engines.

7.4 Triplestores and Business Intelligence

A summary of the differentiating features and advantages of triplestores over traditional data management systems was provided in section 7.3.2. Here we will outline similar advantages and differentiating features, but in the context of typical Business Intelligence related tasks and challenges.

There are several areas related to Business Intelligence where Semantic Technologies and triplestores in particular can provide a significant impact and improvement:

- *Speed up data integration* – data integration and ETL based on RDF is more agile than traditional approaches, since adding new data sources

does not require complex re-design of the existing schemata, which makes triplestores particularly efficient for incremental data integration from disparate data sources.

- *Lower the cost of data integration* – even though the initial cost of using ontologies as enterprise data models is higher, in the long term that cost doesn't increase significantly. On the other opposite side, while traditional ETL has lower cost when the number of operational data sources to be integrated into the enterprise data warehouse is lower, the cost of the process grows with each new data source and is less feasible in the long term.
- *Align and integrate legacy data silos* – querying and consuming disparate RDF data sources is easier with SPARQL as compared to traditional approaches, which makes it possible for applications to deliver timely data to business users even without complex, costly and time consuming ETL processes.
- *Infer implicit and hidden knowledge* – a differentiating feature of the RDF data model, and triplestore respectively, is the ability to derive implicit facts and knowledge from existing data. Many triplestores provide the option for custom, user defined inference rules, which can further enhance the standard logical entailment rules or RDF and OWL.
- *Merging unstructured and semi-structured data* – traditional data warehouses are not particularly suitable for handling large volumes of unstructured or semi-structured data. The simplicity or the RDF graph-based data model makes triplestores good candidates for such application scenarios.
- *Improve the quality of query results* – due to the ability of triplestores to derive new implicit data, and the ability of the SPARQL query language to match data patterns even when the schema of the query terms does not match exactly the schema of the asserted statements, triplestores make it easier to provide better and more up-to-date results when new data is added to the data warehouse.

7.5 Conclusion

Semantic Technologies provide many advantages over traditional data management approaches, which makes them particularly suitable for faster and lower cost data integration and business intelligence within the enterprise. Triplestores in particular provide the means to efficiently integrate, manage and query data in application scenarios where schema agility, low-cost data integration across many data sources and derivation of new implicit information are important.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (2001)
2. W3C Semantic Web Working Group. Semantic Web Case Studies and Use Cases. W3C note (2010), http://www.w3.org/2001/sw/sweo/public/UseCases
3. McBride, B., Klyne, G., Caroll, J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (2004), http://www.w3.org/TR/rdf-concepts
4. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. W3C Recommendation (2004), http://www.w3.org/TR/owl-ref
5. Motik, B., Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Ontology Language Profiles. W3C Recommendation (2009), http://www.w3.org/TR/owl2-profiles
6. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (2008), http://www.w3.org/TR/rdf-sparql-query/
7. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D.: RIF Core Dialect. W3C Recommendation (2010), http://www.w3.org/TR/rif-core/
8. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and Processing. A Collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation (2008), http://www.w3.org/TR/rdfa-syntax/
9. Farrel, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema. W3C Recommendation (2007), http://www.w3.org/TR/sawsdl/
10. Miles, A., Bechhofer, S.: SKOS, Simple Knowledge Organization System Reference. W3C Recommendation (2009), http://www.w3.org/TR/skos-reference/
11. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. W3C Working Draft (2011), http://www.w3.org/TR/2011/WD-r2rml-20110324/
12. Arenas, M., Prud'hommeaux, E., Sequeda, J.: A Direct Mapping of Relational Data to RDF. W3C Working Draft (2011), http://www.w3.org/TR/2011/WD-rdb-direct-mapping-20110324/
13. Provenance Interchange Group website, http://www.w3.org/2011/01/prov-wg-charter
14. Beckett, D., Grant, J.: RDF Test Cases. W3C Recommendation (2004), http://www.w3.org/TR/rdf-testcases/
15. Berners-Lee, T., Connoly, D.: Notation 3 (N3): A Readable RDF Syntax. W3C Team Submission (2011), http://www.w3.org/TeamSubmission/n3/
16. Beckett, D.: RDF/XML Syntax Specification. W3C Recommendation (2004), http://www.w3.org/TR/rdf-syntax-grammar/

17. Berners-Lee, T.: Design Issues: Linked Data (2006), http://www.w3.org/DesignIssues/LinkedData.html
18. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool (2011)
19. Moats, R.: RFC 2141: URN syntax (1997), http://tools.ietf.org/html/rfc2141
20. Paskin, N.: The DOI handbook (2006), http://www.doi.org/handbook_2000/DOIHandbook-v4-4.1.pdf
21. 4store website, http://4store.org/
22. Harris, S., Lamb, N., Shadbolt, N.: 4store: The design and implementation of a clustered RDF store. In: ISWC 2009 SSWS Workshop, Washington DC (2009)
23. Salvadores, M., Correndo, G., Omitola, T., Gibbins, N., Harris, S., Shadbolt, N.: 4s-reasoner: RDFS Backward Chained Reasoning Support in 4store. In: Web-scale Knowledge Representation, Retrieval, and Reasoning (Web-KR3), Toronto, Canada (September 2010)
24. AllegroGraph website, http://www.franz.com/agraph/allegrograph/
25. Bigdata website, http://www.systap.com/bigdata.htm
26. Dydra website, http://dydra.com/
27. Jena TDB website, http://www.openjena.org/TDB/
28. Oracle Database Semantic Technologies website, http://www.oracle.com/technetwork/database/options/semantic-tech/index.html
29. OWLIM website, http://owlim.ontotext.com/
30. Sesame website, http://www.openrdf.org/
31. StarDog website, http://stardog.com/
32. Virtuoso web site, http://virtuoso.openlinksw.com/
33. Kiryakov, A.: Measurable Targets for Scalable Reasoning. LarKC project deliverable D5.5.1 (2008), http://www.larkc.eu/wp-content/uploads/2008/07/larkc_d551.pdf
34. Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. Journal of Web Semantics 3(2), 158–182 (2004)
35. Bizer, C., Schultz, A.: Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints. In: Proceedings of the 4th International Workshop on Scalable Semantic Web knowledge Base Systems, SSWS 2008 (2008)
36. Bizer, Ch., Schultz, A.: Berlin SPARQL Benchmark (BSBM) Specification, v3.0, http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/index.html
37. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP^{2}Bench: A SPARQL Performance Benchmark. In: 24th International Conference on Data Engineering, ICDE (2008)
38. Linked Open Data website, http://linkeddata.org/

8

Service-Oriented Business Intelligence

Alberto Abelló and Oscar Romero

Department of Service and Information System Engineering
Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
{aabello,oromero}@essi.upc.edu

Summary. The traditional way to manage Information Technologies (IT) in the companies is having a data center, and licensing monolithic applications based on the number of CPUs, allowed connections, etc. This also holds for Business Intelligence environments. Nevertheless, technologies have evolved and today other approaches are possible. Specifically, the service paradigm allows to outsource hardware as well as software in a pay-as-you-go model. In this work, we will introduce the concepts related to this paradigm and analyze how they affect Business Intelligence (BI). We will analyze the specificity of services and present specific techniques to engineering service systems (e.g., Cloud Computing, Service-Oriented Architectures -SOA- and Business Process Modeling -BPM-). Then, we will also analyze to which extent it is possible to consider Business Intelligence just a service and use these same techniques on it. Finally, we store the other way round. Since service companies represent around 70% of the Gross Domestic Product (GDP) in the world, special attention must be paid to their characteristics and how to adapt BI techniques to enhance services.

Keywords: Services, Business Intelligence, Service-Oriented Architectures, Business Process Modeling.

8.1 Introduction

As defined in [26], "*services are economic activities offered by one party to another, most commonly employing time-based performances to bring about desired results in recipients themselves or in objects or other assets for which purchasers have responsibility. In exchange for their money, time and effort, service customers expect to obtain value from access to goods, labor, professional skills, facilities, networks, and systems; but they do not normally take ownership of any of the physical elements involved.*" In [40], we find a much simpler way to identify what a service is: The "Unified Services Theory" states that all managerial themes unique to services are founded in customers providing significant inputs into the production process. In [44], it is explained that emerging services emphasize economies of knowledge and adaptiveness,

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 156–185, 2012.

shifting the focus from mass production to mass customization. This shifting is intended to provide superior value by meeting their unique needs for services. Thus, in this work we will analyze the specificity of this sector and present specific techniques to manage and engineer their systems (e.g., Cloud Computing, Service Oriented Architectures and Business Process Modeling). Then, we will analyze to which extent Business Intelligence (BI) can be regarded or even managed as a service and use these same techniques in its engineering methods.

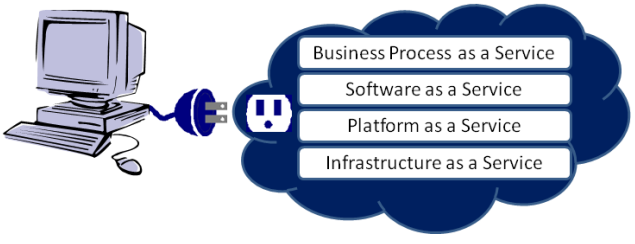


Fig. 8.1. Kinds of services

Services can be provided, mainly, at four different levels. As depicted in Figure 8.1, these are different layers built one on top of the other. Firstly, we can virtualize hardware and provide memory, CPU, disks, etc. This is known as Infrastructure as a Service (IaaS, Section 8.2) and, for example, Amazon's EC2 is a first-class IaaS citizen. Specific benefits of IaaS are fast provisioning and scaling, while the customer still controls the software. As a disadvantage, development and maintenance of applications remains at the customer side. Above this, we find Platform as a Service (PaaS, Section 8.3), where basic software (like Database Management Systems -DBMS- or programming platform) is provided (such as, for example, Google Application Engine). The benefit obtained at this level is that the environment is already installed and periodically maintained, while customer still keeps control of the application. On the other hand, some constrains may appear on what can be installed and porting the application to another service provider may not be easy. At the third level, we can also consume Software as a Service (SaaS, Section 8.4). Now, not only the platform, but also the whole application is installed and maintained by the provider (e.g., Google Apps). On the contrary, now we are tightly tied to the provider and it will be even harder to move to another provider. At the top, we could contract the implementation of parts of our business processes, i.e., Business Process as a Service (BaaS, Section 8.5). In this case, we are externalizing part of the business (not just the software). Well known examples are PayPal, iPhone App store, Chrome web store, etc. Some advantages come now in the form of flexibility, scalability and scale economy, but the relationship with the service provider is of complete dependency.

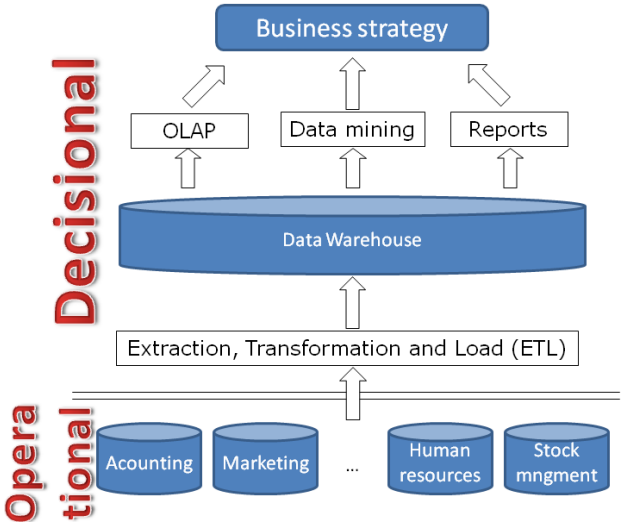


Fig. 8.2. Business Intelligence data flow

Business Intelligence (BI) is the process of making informed decisions in the company. As depicted in Figure 8.2, the central component of a traditional BI system is the Data Warehouse (DW) that stores all relevant historical data in an integrated way. Extraction, Transformation and Load (ETL) tools are in charge of feeding the DW mainly from operational systems or external sources (e.g., through the Internet). On the other side, we find analytical tools that allow users to retrieve data in different ways. The main families of tools are On-Line Analytical Processin (OLAP), Data Mining, and Query and Reporting.

BI systems can benefit from services at any of those four levels in Figure 8.2. In this work, we focus on how to exploit it. In principle, service engineering could not seem feasible for analytical, on-the-fly queries and tasks (e.g., given the amount of data handled by these systems, the highly customizable degree required and the difficulty to outsource processes such as ETL). Furthermore, from the provider point of view, offering BI as a service might be wrongly understood as lowering the barriers to enter the software business (which use to be lower than those in manufacturing, as pointed out in [40]). However, this is not the case. Economies of scale (by easily offering the same service to many consumers through the Internet), capital requirements (for the IT infrastructure needed underneath), and proprietary technology (that can easily evolve) yet fortify those barriers in this case. Indeed, service science happens to be an interesting shot for BI given some of its main characteristics. Interestingly, service oriented systems are described as highly

customizable and user-centered, which suits BI. However, the core difficulties regarding BI service oriented approaches lay on automating the process logic units that would conform the resulting BI system. In section 8.6, you will find a discussion of pros and cons of BI as a Service.

Furthermore, given that service companies represent around 70% of the Gross Domestic Product (GDP) in the world, it is definitely valuable to study and analyze those major drawbacks service oriented systems should face. Thus, we also propose to have a look the other way round and pay special attention to how BI characteristics and techniques can be applied to improve service oriented systems, which is discussed in Section 8.7.

8.2 Infrastructure as a Service (IaaS)

One problem in data analysis tasks is the vast amount of resources they consume. Nevertheless, we do not launch analytical processes day after day or, at least, not of the same size. Thus, a company should provision computer force for the peaks of critical load or for urgent tasks that some day may need. Besides the difficulty to predict when such tasks may arrive and how many resources they would demand, most of the time those provisioned resources will be wasted, because of lack of analytical tasks, or because they just do not need the maximum computer capacity. This waste is directly translated into a loss of money for the company.

A well known e-business, pointed out in [18], is shared IT infrastructure. As defined by NIST (National Institute of Standards and Technology) in [29], IaaS allows for the provisioning of fundamental computing resources (like processing, storage, network, etc.) to deploy and run arbitrary software, which can include operating systems and applications. These services allow a consumer to request and receive a new computer instance without needing to focus on IT concerns such as network placement and hardware availability.

A promising technology for reaching IaaS is *Cloud Computing*, which is defined in [29] as "*a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*". Nowadays, two kinds of clouds are distinguished: private and public ones. *Private clouds* are those data centers that belong to a company and are just used by its departments. On the contrary, *public clouds* are those providers of *utility computing* to the public in a pay-as-you-go manner. In this sense, *utility computing* must be understood as the service sold by public clouds. This perfectly suits the characteristics and needs related to analytical tasks. Instead of buying an expensive machine or data center, we can just *pay-as-we-go*. Thus, throughout this paper, we will use the term "cloud" to refer to public clouds providing utility computing.

Also in [29], we find the essential characteristics of Cloud Computing:

On-demand self-service: Customers being able to provision computing capabilities without requiring human interaction.
Broad network access: Utility computing can be consumed from heterogeneous client platforms (e.g., laptops, PDAs, etc.) anytime and anywhere.
Resource pooling: Computing resources are pooled to serve multiple consumers.
Rapid elasticity: Capabilities can be rapidly and elastically provisioned (they appear to be unlimited from the consumer's point of view).
Measured service: Resource usage can be monitored, controlled and reported (allowing a pay-per-use business model).

Some of these characteristics are emphasized in [48], which, from a hardware point of view, considers that there are three new aspects in Cloud Computing:

1. The illusion of infinite computing resources available on demand (i.e., characteristics "On-demand self-service" and "Rapid elasticity"). In other words, there is not maximum limit.
2. The elimination of an up-front commitment by cloud users (i.e., "Rapid elasticity"). In other words, there is no minimum commitment.
3. The ability to pay per use of computing resources on a short-term basis (i.e., "Measured service").

Specially "Rapid elasticity" and "Measured service" characteristics look like specially interesting for BI, where the load of analytical tasks is not constant or even predictable at mid term. In [40], the authors outline the importance of JIT (Just In Time) information, for which "Broad network access" is a desirable property. Furthermore, it can be better achieved by the elastic power of parallel computing in the cloud (for example, see [50]). Since you are charged by processor cycles being used, you can parallelize tasks as much as possible (you have *"infinite"* available machines), to finish your processing as soon as possible. Provisioning physical machines for the maximum level of parallelism you may ever want to enjoy is clearly unaffordable.

From the economic point of view, Cloud Computing could offer services below the cost of data centers and still make profit thanks, for example, to the difference in the price of energy. A data center can be billed one million euros per year in electricity, whose cost highly varies from country to country depending on the natural resources each has. This cost is mainly due to refrigerating processors, which clearly depends on the latitude of the location. As pointed out in [5], it is cheaper to ship data over fiber optic that to ship electricity over high-voltage transmission lines.

On the other hand, [40] underlines the problem of delaying the delivery of the service, i.e., customers expect to be served shortly after they demand the service. This can be solved by increasing service capacity, which leads to private data centers being clearly underused (studies indicate that they

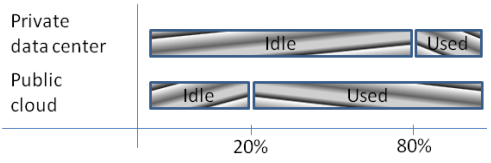


Fig. 8.3. Capacity usage in private data centers and public clouds

remain idle more than 80% of the time), while the usage of a cloud is estimated between 60% and 80% (see Figure 8.3 for a sketch and [5] for details), thanks to multiplexing many tasks. Well known generic strategies for capacity management, as explained in [18], can be used, namely *level capacity* (i.e, *customer-induced variability*, *segmenting demand*, *offering price incentives*, *promoting off-peak demand*, and *reservation and overbooking*) and *chase demand* (i.e., *creating adjustable capacity*, and *sharing capacity*).

In [1], the authors also find Cloud Computing specially appropriate for data intensive applications. Relevantly, it remarks that we only benefit from elasticity if (i) the workload is parallelizable, (ii) data are stored in an untrusted host, and (iii) data are replicated (often across large geographic distances). Going into more detail, ten obstacles (and also research opportunities) for IaaS are detected in [5]. Let us briefly analyze them from the point of view of BI:

Availability of Service: This issue is really important for transactional processing, since a service not available directly affects the number of clients of our company. Nevertheless, for analytical activities this may not be so critical.

Data Lock-In: A problem related to the usage of IaaS is that data must be moved in and outside the provider infrastructure, which usually forces the usage of a proprietary API. If our data sources are not already inside the cloud, we will probably need specific programs to move data in.

Data Confidentiality and Auditability: There should not be any problem to make a cloud as secure and reliable as possible (e.g., anonimization functions and encrypting). Nevertheless, some companies may feel reluctant to give their sensitive data, which, in turn, happens to be the most valuable source of knowledge in most cases. Moreover, some legal issues also have to be taken into account. For example, contracts can explicitly limit the movement of data within national boundaries.

Data Transfer Bottlenecks: Internet transfers are slow. Given that cloud providers charge per data movement (they count bytes transferred) and also the large amounts of data in analytical tasks, this may be a problem (generating extra costs that have to be considered).

Performance Unpredictability: Cloud providers schedule the tasks in their machines, as explained above, in order to level capacity (either memory, disk or any other resources). Thus, if we want our analytical job to be

prioritized it should be so stated in the contract, taking into account variations in the average performance of memory access are smaller than those in the average performance of disc access, for example.

Bugs in Large-Scale Distributed Systems: Debugging is not easy in cloud applications. This may be a problem if we are developing an ad-hoc analytical application, given its complexity and high execution cost.

Scaling Quickly: Given that analytical tasks are data as well as computation intensive, they can present special problems to the provider to absorb their load without violating the service contract.

Scalable Storage: The illusion of infinite resources is easier to be obtained for processors than for storage. In data warehousing, we do not need to scale down, but only to scale up, which simplifies the management and provisioning problem, from the provider point of view.

Reputation Fate Sharing: Laws exist in each country to restrict data management. It should be clearly stated who is responsible (either cloud providers or consumers) for analytical tasks breaking these laws.

Software Licensing: Not only hardware, but also software costs have to be taken into account (charging an annual license does not make sense any more). The provider transfers the cost of the service to the consumer, but it has to be done based on the usage. This problem does not look like having special consequences in the case of BI, beyond those in other kinds of applications.

Finally, according to [2], several challenges appear on having database applications in the infrastructure of a cloud, which are relevant to us, since BI tasks are data intensive:

- Deployment
  - Localization (i.e., generate IP addresses for the virtual machine of each DBMS instance).
  - Routing (i.e., route application requests to the appropriate DBMS instance).
  - Authentication (i.e., be aware of the credentials of all clients independently of where it is run in the cloud).
- Tuning
  - Placement (i.e., mapping virtual machines to physical machines).
  - Resource partitioning (i.e., distribute the resources of the physical machine among the virtual machines running on it).
  - Service level objectives (i.e., be able to minimize resources usage, while maintaining adequate performance).
  - Dynamically varying workloads (i.e., deal with changes in the workload, potentially defining classes and moves from one class to another).

Summing up, the main benefit of IaaS for BI is the rapid elasticity of resources, which is relevant in this kind of applications given the amount of

data and unpredictability of queries. On the other hand, the main problem is that the customer has to trust the provider with his/her data.

8.3 Platform as a Service (PaaS)

As defined in [29], PaaS allows to deploy consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying infrastructure including network, servers, operating systems, or storage (like in IaaS) but platform service offerings include workflow facilities for design, development, testing, deployment, and hosting, as well as services that enable team collaboration, web service integration and marshalling, database integration, security, scalability, storage, persistence, state management, application versioning, application instrumentation, and developer community facilitation. The consumer, in turn, has control over the deployed applications and possibly application hosting environment configurations. These services are provisioned as an integrated solution over the Web.

Cloud Computing does not necessarily lay under PaaS, but it clearly fits. In general, any application or platform can run in a cloud. For example, the authors of [36] contend that we have to start thinking about data management as a service. In the case of DBMSs, it is well known (see [12] and [32]) that the shared-nothing architecture of a cloud easily fits. Nevertheless, some modifications must be introduced to benefit from scalability ([43] notes that differences come from implementation choices and not from fundamental differences in the model). Thus, let us divide the kind of platforms we may find to support BI in two main groups: those made available in a cloud using a special version (normally better exploiting parallelism) or license (i.e., *software to use in a cloud*, which were born outside the clouds and then ported) and those implying cloud and parallel computing underneath (i.e., *cloud software*, which were born to be used in a cloud).

8.3.1 Software in a Cloud

Cloud Computing offers new markets and software vendors are well aware of this. Some examples (not intended to be exhaustive) of tools offering a BI product or version to be executed in Massively Parallel Processing (MPP) systems (i.e., clouds) or just a platform to run or BI development applications are:

- Vertica: A column store, descendant of the open source project C-Store in [42].
- Infobright: An analytic column-oriented database designed to handle business-driven queries on large volumes of data (stored in a grid) - without IT intervention.
- nCluster: Hybrid row and column DBMS.

- K2Analytics: A Managed Service Provider (MSP), which houses the customers licensed software.
- LogiXML: A code-free application development environment including hundreds of pre-built elements, that reduce the amount of work needed to create and maintain more complex BI applications.

8.3.2 Cloud Software

Despite any application can run in a cloud, some projects appeared lately that propose new architectures and algorithms to exploit its specific characteristics and possibilities. Systems in this category use to be tagged with the "NOSQL" buzzword (standing for Not Only SQL). Nevertheless the difference is not whether they provide an SQL interface or not, but the data model they follow, which is not relational. Thus, the temptation might be to name them "NORelational". Nevertheless, as explained in [28], "CoRelational" would be more appropriate, as they could be better understood as complementary.

These systems are based on the storage and retrieval of key/value pairs (as opposed to the relational model based on foreign-key/primary-key relationships). Relational referential integrity assumes a closed world (where all data are known, as opposed to an open world), which is hard to be assumed in a cloud or the Internet. While in the Relational model child nodes point to the well known parent (by means of a foreign-key), in the CoRelational model are parents who point to their children (whose information may not exist beyond the parent limits). Thus, instead of having each sale pointing to the corresponding product, date, customer, etc., we keep a lists of pointers indexed by the product, date and customer key (i.e., we reverse the pointers). It is in this sense that [28] compares both models under the prism of *category theory* to conclude that they are actually dual.

In the following subsections we just summarize the main concepts of BigTable and MapReduce as they were presented by Google in [9] (open source implementation [4]) and [11] (open source implementation [3]), respectively (another interesting Google product, we are not going to analyze, is Dremel, for details of the latter see [30]).

BigTable

is a distributed storage system designed to scale to very large size (petabytes). Figure 8.4 sketches data organization inside it, whose main structure is [key,value] pairs. The main characteristics we would like to outline here are:

- Data are indexed by row and column values (which are arbitrary strings).
- Columns can be grouped into families to be physically stored together.
- Versions are automatically generated for each value (which are timestamped).
- Data are treated also as uninterpreted strings.

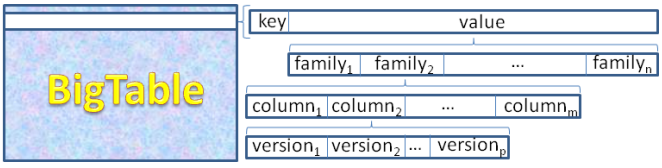


Fig. 8.4. BigTable organization

- Only single-row transactions are supported.
- Data are clustered (i.e., physically sorted) by key.

Note that only families of columns are part of the schema and have to be stated on creating a table. Oppositely, the columns are dynamically defined on inserting data. A family actually corresponds to a separate storage. Regarding data retrieval, it provides random access to one key, as well as parallel scan of the whole table or a range of keys.

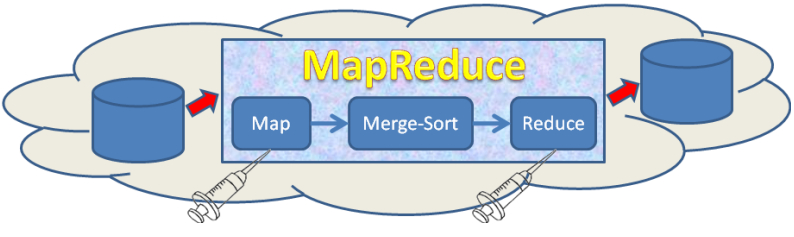


Fig. 8.5. MapReduce overview

MapReduce

Is a programming framework that allows to execute user code in a large cluster. It hides parallelization, data distribution, load balancing and fault tolerance from the user. All the user has to do is writing two functions: Map and Reduce. As sketched in Figure 8.5, those functions are injected in the framework.

Thus, the signature of those two functions is as follows:

$$map(key_{in}, val_{in}) \rightarrow \{[key^1_{tmp}, val^1_{tmp}], ..., [key^n_{tmp}, val^n_{tmp}]\}$$

$$reduce(key_{tmp}, \{val^1_{tmp}, ..., val^m_{tmp}\}) \rightarrow \{[key^1_{out}, val^1_{out}], ..., [key^p_{out}, val^p_{out}]\}$$

The execution would be:

1. Map function is automatically invoked for each pair [key,value] in the source table (it also works for plain files, in which case, one pair is generated per line, having as key the position inside the file). Notice that the source table is distributed in a cloud. Therefore, the framework takes advantage of this and tries to execute each map call locally to the data. Each call can generate either one new pair (potentially different from that in the input), many pairs or none at all.
2. The intermediate pairs [key,value] generated by the different executions of the Map function, which are temporally stored in the distributed file system, are then ordered using a distributed merge-sort algorithm.
3. For each different intermediate key, the Reduce function is invoked once receiving together all values associated to it. Each call can generate again either one new pair (also potentially different from that in its input), many pairs or none at all.

In [35], the authors compare and analyze the performance of Hadoop (which is an open source MapReduce implementation) against that of two parallel DBMSs. They conclude that DBMSs outperform Hadoop, while Hadoop is more tolerant to software/hardware failures. Reasons for these differences are carefully analyzed and justified to conclude, as in [43], that both technologies are rather complementary. [23] identifies five design factors that affect the performance of Hadoop and shows how by tuning these factors its performance improves to be almost comparable to that of the DBMSs. Another problem of working in a distributed environment (already pointed out in previous section) is that of debugging. KarmaSphere offers a development environment for Hadoop that allows to monitor the execution and debug your code.

Generalizations of MapReduce are presented in [46] and [6]. The latter extends the framework to allow other kinds of operations (i.e., Map and Reduce are just special cases of the operations that can be injected by users). Special attention is devoted to operations with more than one data input stream (e.g., join, which results to be really hard and artificial to be obtained in MapReduce).

Both technologies fit naturally to advanced BI tasks. For example, [21] argues that next data warehouse generation must consider all kind of data available, which includes textual data (not only plain text files but others such as mails) and unstructured sources (generally coming from the Web). The immediate consequence is that the amount of data to be stored grows exponentially and accordingly do so the size of the analytical tasks to be performed. BigTable and MapReduce are two promising paradigms to tackle, respectively, both consequences. Whereas BigTable scales to petabytes, the MapReduce paradigm transparently (i.e., the framework is responsible) provides grouping and ordering (essential for analytical tasks) regarding the key values produced.

All in all, the benefit of PaaS is that it hides the complexity of managing parallelism, which is mandatory for "right-time" BI. Indeed, systems like MapReduce were specifically conceived for analytical tasks.

8.4 Software as a Service (SaaS)

As defined in [29], SaaS allows to run the providers application. The consumer does not manage or control the underlying infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. Software services allow a consumer to select a specific software instance without the need to be aware of where and how it is hosted. For example, a developer can request a word processor instance (i.e., Google apps) without having to be aware of which OS or hardware the application will run on. This allows the consumer to focus on the characteristics of the application and gives to the provider the freedom to fulfill the request with any resources that meet the need.

These are services that provide remote (e.g., Web-based) access to software that is actually running in provider's machines. As pointed out in [5], both (i.e., consumers as well as providers) benefit from this architecture. On the one hand, consumers gain access anytime and anywhere, and can easily share data and collaborate. This can bring huge benefits from the BI perspective, as data analysis could be carried out or studied on-the-fly with portable devices such as smart phones. Moreover, the pay-as-you-go model is also more beneficial than licensing where you pay per number of CPUs or users, for example. You only pay for the software if you really need to analyze your data. On the other hand, providers simplify software installation and maintenance, and centralized version control.

As explained in [20], the percentage of today revenue from BI as a service offerings is insignificant compared to the overall BI platforms market (less than 5%). [20] only mentions SAP/Business Objects "On Demand" as a product with a significant presence. Nevertheless, the penetration in the market depends on the specific kind of applications, and some are more mature than others. Gartner's report shows that SaaS versions of ERP (Enterprise Resource Management) are much less popular (below 1% of all ERP) than those of CRM (Customer Relationship Management, e.g., *Salesforce.com* or *Cloud9 Analytics*) or SCM (Supply Chain Management, e.g., *Oco*) systems (around 12% and 18% respectively). Examples of other BI software offered as a service are reporting, dashboarding, etc. (e.g., *Microstrategy*, *Quantivo*, *Panorama* or *ColdLight Neuron*). Let us analyze the former in the following sections.

8.4.1 Supply Chain Management (SCM)

As explained in [25], SCM systems enable the firm to model its existing supply chain, generate demand forecast for products, and develop optimal sourcing and manufacturing plans. They allow to exchange and share information between partners (i.e., producers and consumers) in the supply chain. SCM allows to minimize the *Bullwhip effect*, which is the distortion of information about the demand for a product as it passes from one entity to the next across the supply chain (i.e., little changes in the demand of the final consumer generate huge variations in the raw materials).

In the beginning, these systems where developed for each company. Afterwards, Electronic Data Interchange (EDI) appeared and allowed to efficiently exchange documents like orders or bills. However they yet did not help to manage the process. Nowadays, SCM systems allow a pull-based (i.e. build-to-order) model, where the customer order triggers events in the supply chain, so that production, inventory and shipping can be easily adjusted to real demand. In the future, the Internet could allow that all companies in the supply chain can see what others (not only immediate providers and consumers in the chain) need and generate at any time. This would facilitate not only fast response but also more precisely forecasting consumption.

However, this is not easy to achieve, because requires several organizations to share information which is an organizational, as well as a technological challenge. From the technological point of view, services (and service architectures) can be really helpful at this point.

8.4.2 Customer Relationship Management (CRM)

The concept of CRM appeared with this century. It can be defined as having an integral view of each and every client in real time. The goal is to be able to make fast, informed decisions regarding marketing strategies or competitive positioning (i.e., differentiate your offering and create value for your customers). This is achieved by tightening seller-buyer relationships.

Despite CRM has a big technological component, it is usually seen as a way of doing things in the company. As explained in [25], a CRM system captures and integrates customer data from all over the organization, consolidates and analyzes it, and distributes the results to various systems and customer touch points across the enterprise. Nevertheless, hardware and software must be accompanied by the definition of the corresponding processes or workflows, objectives and indicators.

To achieve these goals, the whole company must be involved in the deployment of the CRM. If such change generates more costs than benefits (specially to the marketing department), it will hardly succeed. Some web-based CRM software systems exist that can be licensed for a few hundred euros per user per month. In front of this, buying, installing and personalizing the technology can cost millions of euros to the company. Thus, Cloud Computing and

a service approach makes feasible for small companies to implement a CRM, or at least try to do it reducing the risk in case of failure of the project.

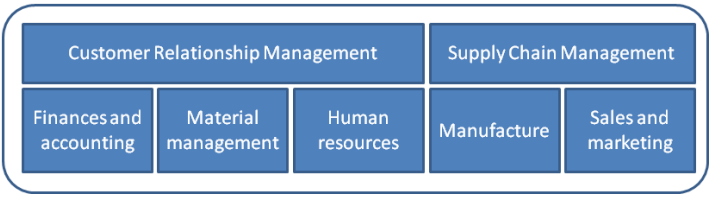


Fig. 8.6. Typical ERP modules

8.4.3 Enterprise Resource Planning (ERP)

Historically, enterprises have deployed independent systems to manage different processes in the company, which leads to, not only separate software and hardware, but also separate data that eventually have to be merged and integrated to make decisions. ERP systems try to solve this by providing one common repository where data are collected from all business processes. As depicted in Figure 8.6, different modules are available implementing common processes in the enterprises (e.g., Finances and accounting, Material management, Human resources, Manufacturing, Sales and marketing, etc.), and they just have to be properly configured and tuned up to behave as close as possible to the way a specific company runs the business. This helps not only the business processes themselves, since they share the information, but also facilitates decision making (most ERP systems have SCM and CRM modules).

Acquiring a software package always makes your company dependant on the software provider (obviously for installation, but also for upgrading and maintenance), and switching from one provider to another may not be easy. This is specially true in the case of ERP, because such systems drive all the operation of the company. Indeed, things are even worse if the ERP is contracted as a service, because the provider also hosts the data (special care must be taken at this point to guarantee that we may move our data out of the system if so desired). Upgrading and maintenance are much easier (and probably cheaper), but security issues pop up. SAP Business Suite, Oracle's e-Business Suite, Microsoft's Dynamics suite, and IBM's WebSphere are examples (and use Web services and SOA inside).

8.5 Business Process as a Service (BaaS)

Nowadays, the rapid changes in society, in general, and business in particular make enterprises an ever-evolving entity. Accordingly, applications and information systems are required to evolve quicker than ever and match the organizations needs. Special techniques are necessary to support this, allowing IT

to evolve at the same pace as business requires. One promising approach is to divide the processes in our business into units of processing logic. These units are collections of units of work that can be provided as a *service* (either from inside or outside the company). A service hence provides a specific capacity for the user and ideally, these units should be highly both customizable and reusable in such a way that complex systems could be seen as a myriad of them interacting as a whole.

A promising paradigm to tackle this scenario are Service-Based Applications (SBAs), see [34]. Software services constitute self-contained computational elements that support flexible composition of loosely coupled distributed software systems. This implies fundamental changes to how software is developed, deployed and maintained. The relevance of software services is increasing given the crucial role played by the Internet: on the one hand data distribution is a fact for most real organizations (either naturally distributed by geographic criteria or artificially like in a cloud to exploit parallelism). On the other hand, global connectivity brings services to another level, as they can be thought as pervasive, ubiquitous and highly dynamic units which could be searched, browsed and listed. Indeed, a clear example of SBAs can be found in the current *App stores* available for smart phones and tablets, which are nowadays already available for browsers and, in turn, for laptops and desktop computers.

Indeed, nothing prevents us to generalize the concept of SBAs not only to applications but to whole information systems inter and intra-organizations. According to the NIST Working Group, BaaS focuses on providing existing business processes through a cloud. If there is an existing process whose specifications are known, it can be provided as a service within the catalog. This allows the service provider to automate any steps within the process while leaving the changes transparent to the service consumer. There already are some examples in this direction, and it is rather usual to find outsourced paying methods in the Web (e.g., PayPal or Moneybookers) or geolocation stuff (e.g., by combining the smart phones built-in GPS features plus Google Maps).

BaaS, however, introduces new challenges and specific techniques to engineer service-based systems (e.g., Service Oriented Architectures and Business Process Modeling) have emerged lately. According to [34], we can distinguish between challenges related to service technologies (i.e., the means) and service principles, techniques and methods to develop service-based systems. The main challenges regarding service technologies are the following:

Service Infrastructure: It refers to basic infrastructure on top of which the service-based systems are built. It refers to the most basic service layer responsible for communication primitives and patterns, architectural constructs to connect heterogeneous systems, service access and a runtime environment for services execution. It is also responsible for service search, identification and publication. In the context of this paper, the first two previously discussed service layers (i.e., IaaS and PaaS) could sit below

BaaS and provide the needed service infrastructure to built on its top service-based systems.

Service Composition and Coordination: Service oriented systems loosely couple logic units (i.e., services) to aggregate them in an interoperable way and produce a single, complex service. SBAs are built on Service Oriented Architectures (SOA). An important property of services taking part in an SOA is their composability (see [10]). There are several options to service composition such as service orchestration, or service choreography among others (see Section 8.5.1).

Business Process Management (BPM): BPM can be defined as mechanisms to understand, capture, represent and manage organizations, and is aimed at developing end-to-end business processes. Its main objective is to fill the gap between business processes and IT applications, to better understand an application's requirements. It can be understood as a management principle, but also represents suites of software aimed at bridging IT systems and people (see [17]). Current SBAs typically involve well-defined processes (such as payment processing, shipping, tracking, etc.) which must be understood as a whole. However, managing a finer-granularity (e.g., business data, events, Quality of Service -QoS- agreement, Key Performance Indicators -KPIs-, etc.) is needed to guarantee a cohesive and continuous information flow, at the level of business design and not merely at technical level. Business Rules Management Systems (e.g., Drools or IBM WebSphere ILOG JRules) are a natural partner for BPM systems (like IBM BPM tool), as they allow to store the business decision logic in a centralized repository.

On the other hand, on top of service technologies, service principles, techniques and methods focus on developing service-based systems. Basically, [34] highlights the relevance to come up with a specific lifecycle for service oriented systems. Indeed, it cannot be assumed that existing applications can be imported into an SOA by merely creating wrappers and leaving the underlying component untouched. A service-based development method is required to specify, construct refine and customize highly flexible service-based systems. For example, the adapting and evolution of services based on monitoring performance indicators and an automatic Quality of Service (QoS) negotiation between services are reasons why the life-cycle must be extended. Thus, service-based systems are not just a thin SOAP/WSDL/UDDI layer on top of existing systems or components. In short, new challenges (regarding traditional systems) about developing service-based systems can be summarized as follows:

Service Engineering and Design: This should provide specifications for composition and coordination of the services forming the deployed system, which guarantee it does not produce spurious results. The process should lead to high-quality service-based systems. The BPM layer would also be included in this item.

Service Quality: Service quality definition, negotiation and assurance (ideally in an automatic way) is a key aspect for service-based systems at all levels. This includes assessing a service quality level both for controlling the agreed communication quality level between services forming the system and for agreeing on the quality of the output generated. Relevantly, end-to-end quality provision implies to propagate quality control on attributes from the service infrastructure, service composition and business process management. Therefore, it is transversal to the whole system.

Service Adaptation and Monitoring: Tightly related to the previous item, service-based systems should adapt and evolve to answer contextual changes and even predict upcoming problems. In this sense, monitoring, predicting and governing the activities of the service-based system emerge as key aspects.

For these reasons, as discussed, an entire specific service lifecycle (identifying, finding, designing, developing, deploying, evolving, quality assuring and maintaining services) must be considered. In the remainder of this section, we will focus on two specific concepts related to BaaS from which BI can vastly benefit. Firstly, we analyze how SOA can be used as an architecture for providing BI, and then how to define, negotiate and assure the quality of the BI services.

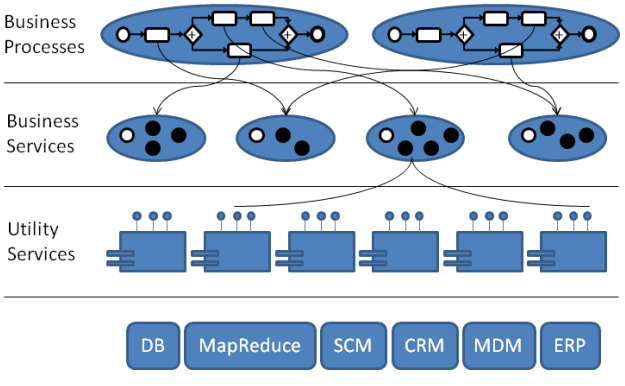


Fig. 8.7. Service Oriented Architecture

8.5.1 Service Oriented Architecture (SOA)

As defined in [15], SOA is a term that represents a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units comprise a larger piece of business automation logic, namely processes or activities (see Figure 8.7). Individually, these units can be distributed. Web services, including standards such as WSDL (Web Services Description

Language), SOAP (Simple Object Access Protocol) and WS-BPEL (Web Services - Business Process Execution Language) are the most popular realization of SOA (see [10]).

Three overlapping kinds of services exist:

Business service: It represents the most fundamental business block and encapsulates a distinct set of business logic. At this level, Business Process Modelling (BPM) is a natural complement to SOA towards the objective of aligning technical initiatives with strategic goals of the business (see [33]).

Utility service: This provides generic services designed for potential reuse (black circles in Figure 8.7).

Controller service: It is responsible to coordinate service-to-service composition members (white circles in Figure 8.7).

Compared to two-tier client-server architecture, SOA offers more than one decomposition layer (i.e., a service can be successively decomposed into other smaller services). Moreover, processing units are much smaller than typical monolithic client or server applications. Multi-tiered client-server applications do not necessarily follow SOA, even if they are distributed and incorporate Internet technology, among other reasons, because the components are still tightly coupled. Even using Web Services in these applications does not entail SOA, because they do not guarantee the independence between components.

Thus, eight common principles to SOA are stated in [15]:

Reusability: Services are designed to support potential reuse.

Loose coupling: Services must be designed to interact without the need for tight, cross-service dependencies.

Contract: They need not to share anything but a formal contract that describes each service and defines the terms of information exchange.

Abstraction: The only part of a service that is visible to the outside world is what is exposed via service contract.

Composability: Services may compose other services.

Autonomy: The service does not depend on other services for it to execute its governance.

Statelessness: Services should be designed to maximize statelessness even if that means deferring state management elsewhere.

Discoverability: Services should allow their descriptions to be discovered and understood by human as well as service requestors that may be able to make use of their logic.

Each one of these characteristics emphasizes one different aspect of services. However, all of them pivot around the concept of reusing software as much as possible. Thus, in order to reuse, we must also be able to compose different services. The interaction of a group of services working together to complete a task can be referred to as a *service activity*. This, as explained in [15], can also be achieved in some different ways:

Primitive activity implements synchronous communication between two services. It is short-lived, and typically involves only one message exchange between them.

Complex activity involves many services and messages exchanged among them, that allow to carry out complex tasks. It usually spans a longer period of time and requires some coordination framework.

- Atomic Transaction wraps a series of changes into a single action (usually ACID-compliant). Its internals provide some protocols to guarantee it (e.g., two-phase commit). Atomic transactions play an important role in ensuring service quality and lead to a robust execution environment.
- Business Activity participants are required to follow specific rules defined by protocols. Oppositely to an atomic transaction, it spans hours, days or even weeks. Moreover, business activities provide an optional compensation process that can be invoked when exception conditions are encountered (i.e., they are not expected to rollback any changes). An added difficulty for this kind of activities is that they cannot expect to retain the participants for the duration of the activity, since the coordinator does not have any control over them.
  - Orchestration participants interact with a central engine, which provides the workflow logic. This logic can be expressed, for example, in BPMN (Business Process Modeling Notation) and BPEL (Business Process Execution Language),see [49].
  - Choreography is intended for public message exchanges, so that collaboration between services can be implemented without needing the central coordinator of the workflow logic. This is achieved by assigning roles to the participants. Choreographies are conceived to achieve collaboration among services from different, independent providers.

Using BPM to model the whole BI process definitely makes sense, since BI processes are completely business-oriented and can be modeled as workflows (specifically, as data transformation workflows). However, it seems reasonable to only outsource (i.e., contracting external services to carry out the task) certain BI tasks within BI processes. For example, contracting additional services providing input data for the BI process (e.g., stock markets data or news feeders) or for providing external rules used in the process to transform data (e.g., spam detecting and mail filters for data cleansing or external dictionaries / ontologies to support data integration) fully make sense, but it would not be reasonable to outsource intermediate tasks within a business process, as it would provoke massive data shipping between services (as discussed, Cloud Computing entails that we are charged according to the amount of data shipped). For example, we may contract input data (such as the stock market daily data), load it into a service-based system and carry out as many transformations as needed to match up the BI process goals. However, shipping data to external (outsourced) intermediate services in the middle of the process and send it back after some transformations is

not reasonable cost-wise. Following this idea, any BI process can be seen as data transformation from the data sources to the Key Performance Indicators (KPI) exploited by the managers. Data manipulated all over the process is transformed in two different ways: (i) dynamic on-line transformations (e.g., OLAP engines), which allow the users to query sources (such as decisional databases) and produce data analysis on-the-fly, and (ii) static off-line transformations, which would match the traditional ETL process to produce the sources queried dynamically and on-line by the user (i.e., to populate the decisional database). Given this classification, SOA seems more promising for the latter, as discussed below.

In [47], the framework offers an extensible palette of template activities for ETL processing. These activities are defined there as an amount of work which is processed by a combination of resource and computer applications. Moreover, this activity can be performed in a black-box manner. Note that the same word "activity" is also used in [15] to refer to sets of coordinated services. Thus, each ETL flow can be considered as a separate business process. Then, each business process would be implemented as a complex activity. Some would be atomic transactions able to rollback to a safe state. However, most of them would be business activities, demanding the definition of a compensation process to leave data consistent in case of a not completely successful execution. These activities would be defined in the form of orchestrations. There are works (for example, [13]) that already propose the usage of an SOA architecture, where different data mining elements interact to complete the analysis of data (financial market in this case). The characteristics of services would allow to reuse components or just change the provider (if we find another one improving the quality of the BI process).

As explained in [8], we can model the ETL workflow as a business process. This would allow to have a conceptual view that can be mapped later to the implementation. It would hide low-level IT events resulting to be more comprehensible to analysts. In some cases this would result really helpful for business managers to, not only understand, but also define how data have to be transformed.

We find in [34] a list of major challenges for the near future of services. Among them, we find some that would clearly be useful to BI, like "Infrastructure support for data integration" and "Semantically enhanced service discovery". On the other hand, it also outlines the current lack of tools for supporting the evolution and adaptation of [business] processes, so that it is hard to define compositions of distributed [business] processes that work properly under all circumstances.

8.5.2 Quality of Service (QoS): Definition, Negotiation and Assurance

With services, quality "specifications" come from multiple simultaneous sources, including the company and the individual customers. The company presents specifications as standard operating procedures. The customer

presents specifications based on their need-driven expectations for changes to their process-inputs. Misalignment between company- and customer-specifications for the service process leads to dissatisfaction, even when the process goes exactly as it was designed. The misalignment of specifications can be avoided through communication. However, if the service performance does not address individual customer needs, the customer will not require the service. With services, expectations are often subjectively acquired and subjectively defined. Therefore service providers need to be careful when attempting to define expectations for customers. Too low expectations can lead to loss of sales. Too high expectations can lead to disappointment and lost future sales [40].

With this spirit, service-based systems must negotiate and agree quality aspects regarding the system. Service providers need to characterize their services to define both the offered functionalities and the offered quality. For example, according to [37], these quality aspects may embrace quality attributes regarding runtime, transaction, configuration management and security features. Furthermore, not only the need to define, negotiate and agree the QoS is growing, but also its eventual validation and verification at run-time.

As stated in [34], QoS is measured by the degree to which applications, systems, networks, and all other elements of the IT infrastructure support availability of services at a required level of quality under all access and load conditions. As discussed in [44], measuring and evaluating performance remains a difficult problem. A service should be developed and delivered to achieve maximum customer satisfaction at minimum cost. To facilitate this task, they provide a tentative list of measures partitioned in four categories to evaluate the service:

Categories	**Example Evaluation Measures**
Input	Demand
	Supply
	Cost
Process	Performance (Quality, Reliability, Speed, Throughput)
	Satisfaction (Efficiency, Effectiveness)
	Safeguards (Privacy, Security, Safety)
Outcome	Customization
	Satisfaction
	Convenience (Availability, Accessibility)
	Robustness (Comprehensiveness, Adaptability, Flexibility)
Systemic	Consistency
	Equity
	Reproducibility
	Sustainability

The input and process measures serve to explain the resultant outcome. Input measures indicate the potential of our system and what it needs to run (i.e., they measure how the input looks like; e.g., demanded input or

cost), whereas process measures (e.g., speed, efficiency, privacy, etc.) consider how the activity is performed but not the impact it has. This is the main reason for the third set of measures, which focuses on measuring the ultimate results (e.g., its degree of customization or the user satisfaction). Finally, the systemic measures are regarded as impact measures over other systems or services (e.g., sustainability or consistency).

As shown in [14], data intensive applications can benefit from an agreement in the QoS in advance to the actual data retrieval. A Service Level Agreement (SLA) is based on a set of measurable characteristics of a service known as Service Level Objectives (SLO) (e.g., price, availability, response time, number of available tuples, number of retrieved tuples, query cost, locality and legal issues, etc.). Note that SLOs of complex activities must be decomposed for the participants in the activities. The other way round, estimation of each characteristic must be aggregated for the services composing such activities.

As pointed out in [8], implementing the ETL workflow as a business process would also help to manage its monitoring and reporting. In this sense, another important challenge is the implementation of "QoS-aware service compositions" that takes into account the performed data transformations (see [27]).

8.6 Discussion

This section contains an overall discussion of advantages and disadvantages of using services (at any of the four above mentioned levels) for BI. First of all, one key aspect regarding a service is that it entails outsourcing (in general, from the company, but also just from the department or business unit). Roughly speaking, it allows the company to spend the efforts in making decisions instead of building decisional systems. Before other considerations, thus, we must take into account that, as pointed out in [18], outsourcing has benefits and risks from the customer point of view. The list of these benefits can be summarized as follows:

- Allows the firm to focus on its core competence.
- Decreases cost by purchasing from an outside source rather than performing in-house.
- Provides access to latest technology without investment.
- Leverages benefits from a supplier who has economies of scale.

Oppositely, main risks to consider are:

- Loss of direct control over quality.
- Exposure to data security issues.
- Dependence on one supplier compromises future negotiation leverage.
- Additional coordination expense and delays.
- Atrophy of in-house capacity to perform outsourced services.

Thus, the key question is how to guarantee that the selected service does really match our needs. In other words, how to minimize the outsourcing risks and maximize its benefits. In [18], the authors highlight some criteria that might be used to select a service provider:

Convenience: Some locality issues may be considered due to legal conditions or connectivity.
Dependability: Is the provider reliable?
Personalization: Customization is an important, not to say crucial, characteristic in any service, as it must adapt to our needs.
Price: Obviously, the cheaper (provided a given set of features), the better.
Quality: Defined as a function of previous expectations and perception of the experience.
Reputation: Previous customers' experiences are important.
Speed: How long it will take to complete the process.
Security: In the broad sense of CIA (i.e., Confidentiality, Integrity and Availability), as pointed out in ISO/IEC 17799, is a must.

All these criteria happen to be yet relevant for BI but, specifically, security, quality, and personalization shall draw the attention of BI customers. Indeed, regarding BI, security is a major concern for them, since the (potentially sensible) data of the company has to be transferred, managed and even stored by service providers, raising, consequently some inherent questions such as: Are my data safe? Will data be available when needed? Moreover, the provider must guarantee that data are not only geographically distributed, but also replicated to reduce their vulnerability to catastrophes and network partitions. Regarding quality, BI processes should guarantee a minimum threshold of quality previously negotiated and agreed with the provider. Furthermore, monitoring services to check whether the agreed quality level is reached or not should be available for the user. Finally, how to customize, adapt and evolve the available services to fully match customers' specific analytical needs is a challenge for BI providers, since by definition, analysis requirements will be different for each company.

On the other hand, the advantage of externalizing is that you pay for what you really need at every moment. This solves the problem of underusing the customer's data center, but from the provider's viewpoint, a capacity management problem appears to deal with variations in the demand. Given that analytical tasks are data as well as computation intensive, they can present special problems to absorb their load without violating the service contract.

Nevertheless, you should note that the benefit at the customer size is huge, because (if using a scalable platform) the result of a computation can be obtained in as few time as desired (by just hiring more parallel resources) without a substantial increment in the overall cost. The problem the customer has to face is that of transferring data to the providers side (if they were not already there). Internet transfers are slow and data movements in and out of

a provider are usually charged. This can make the difference given the huge amount of data in BI applications.

Leaving aside the economical issue of moving data in and out of the provider's machines, a service architecture allows to face one of the most challenging BI problems we have today, i.e., the elastic capacity provides the flexibility to analyze in "right-time" the huge amount of unstructured data flowing through the Internet (e.g., e-mails, twits, reviews, comments, evaluations, etc.). The challenge is triple: we firstly have to deal with a humongous amount of data; these data are not structured; and finally, we want to make a decision as soon as possible to avoid missing a business opportunity.

Also from the customer point of view, by hosting on one machine the BI applications used by many companies, licenses for background software only have to be paid once, and the software code updates are immediately available for all users, who have access to their data anytime and anywhere. However, this raises the problem of lack of dedicated IT personnel (i.e., atrophy of in-house capacity) and loss of access to the backstage. The end user should be self-sufficient. Some recent works go in this direction, either helping on the management of the operating system [22], the DBMS [7] with a DW workload [16], the ETL processes [8], the multidimensional design [38, 39] or the queries posed [19, 24].

8.7 Business Intelligence on Services

In the previous sections we have discussed how BI can benefit from service science, whereas in this section we focus on the other way round: how services can benefit from BI characteristics and techniques and how to apply them to improve service-oriented systems.

It has already been shown how BI might help to turn traditional manufacturing into services. Basically, service principles (such as customization, adaptiveness, quality negotiation, etc.) can be applied by exploring the available data of the organization and extract relevant knowledge, which will allow us to, for example, adapt our products to customers. To undertake this process, BI techniques are of great relevance. As pointed out in [18], in contrast to manufacturers, knowing the relationships with customers is a significant competitive advantage in service companies (since they are co-producers of the services). Thus, CRM (as explained in Section 8.4) is really important in this sense to the point of even becoming a barrier to entry for competitors. These systems are used to, based on how profitable their business is, code customers with instructions for service staff, route them in call centers to place them in different queues, target the offers and share their data with other firms. The problem here is that we must track all details about customers, resulting in a huge volume of data, but again, this is the main objective of traditional BI systems such as Geographic Information Systems (GIS) or data warehousing.

Relevantly, by allowing customization of their products manufacturers gain service characteristics. In this case, it is specially important to coordinate the whole chain from parts suppliers to retailers. Thus, SCM (see Section 8.4) represents a competitive advantage as well as a challenge because of the network model we have in general, and the level of uncertainty.

Furthermore, making decisions to improve service quality is also a must. Quality is a differentiation strategy to gain competitive advantage. Nevertheless, as stated in [40], it tends to be subjective and difficult to scale, because it is defined based on expectations and perception of the service. Nevertheless, this difficulty does not impede that KPIs are usually defined for the different processes of the companies.

All in all, information, understood as the result of processing, manipulating and organizing data in a way that adds new knowledge to the organization receiving it, is the keystone of service-based systems. Consequently, databases are a great asset to service companies and, for example, they can sell information about their customers (if legality allows it) or can directly use it for marketing purposes. At this point, traditional BI techniques such as reporting, OLAP and data mining play a critical role. In the following, we will focus on how services monitoring can be taken to another level by applying BI techniques and exploiting the derived knowledge for assuring the quality of services, allowing them to evolve and adapt and, in short, make them perform more efficiently.

8.7.1 Monitoring Our Services

Monitoring is a key aspect to guarantee the success of service-based systems. Monitoring refers to the need to monitor the business processes in order to (re)-design, adapt and evolve them (see [41]). Tightly related to BPM, an example of software covering this area is IBM Business Monitor. Most companies already have a tool (commonly known as "Balance scorecard") that allows to follow the consequences of the decisions made and easily check whether they succeed or fail. The main idea behind monitoring is that you cannot control what you cannot measure. Therefore, relevant knowledge must be available in order to make better decisions. Traditionally, a set of KPIs are defined and showed in a graphical way (e.g., balance scorecard, digital dashboards or any other Performance Measurements System). To do so, normally data are first gathered and integrated in decisional databases which are later used to nurture dashboards, balance scorecards, etc.

In its broader sense, monitoring has successfully been applied in BI for a while (see [8]). The innovative aspect in service-oriented systems, is that Key Performance Indicators (KPIs) are tightly related to business processes from the very beginning at the definition and design steps. Indeed, managers and decision makers could define them in high level languages such as BPEL and BPMN and make them flow all over the technical aspects bridging the gap. As an example following this principle, Figure 8.8 presents IBM's Websphere

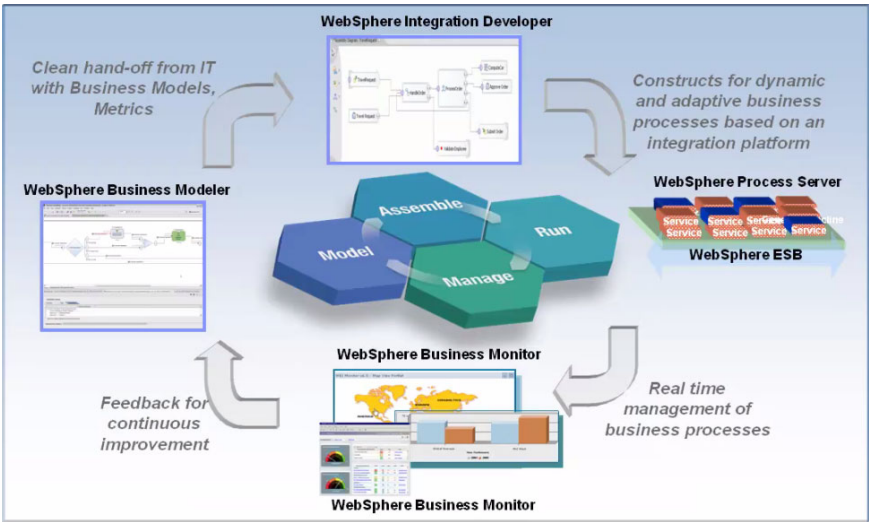


Fig. 8.8. IBM WebSphere BPM components

components, which follow this principle to let the system evolve and adapt to the customer needs. In this framework, the manager would firstly define the process using a BPMN diagram. Then, this is automatically passed to the developers that have to implement and assemble the corresponding service components. Those components are executed in a runtime environment, than can be interactively monitored and modified.

However, KPIs cannot be only conceived as performance indicators at the client end but also at the provider end. Indeed, they can also be internally applied to measure the quality of services. For example, we can define KPIs to measure if the Service Level Objectives (SLOs) are reached (and thus, if the provider - customer agreement is fulfilled). These KPIs do not directly impact on the business processes implemented but on the quality of the selected services. This valuable knowledge can be further exploited to adapt and evolve our SLOs and eventually, our service agreements. In this sense, [8] automates the design, implementation and adaption of ETL for business processes, by recording and monitoring their events. It should be during the definition of this ETL process that the KPIs of the company are defined. Indeed, in this work, the authors also explain how the definition of KPIs is also facilitated in their framework.

As pointed out in [45], it is really interesting to establish the correlation between IT performance and business performance metrics (i.e., KPIs). Thus, metrics (beyond failure, availability and response time) should be defined to data processing and see how these affect the business. Importantly, many data are generated in BPM systems. [31] classifies these data and analyzes

their complexity in terms of volume and variance. Obviously, the more data we have and more they vary, the harder it will be to analyze them. At this point classical BI techniques such as reporting, OLAP and data mining become essential to enable successful data analysis. Reporting is of great value to periodically trigger well-known analytical processes to control workflow processes, whereas OLAP and data mining provide means to facilitate advanced analytical tasks based on KPIs. OLAP principles allow us to easily aggregate performance indicators and analyze their values according to different perspectives of analysis, whereas data mining provides algorithms to derive and infer hidden knowledge based on statistical data transformations. All in all, BI techniques become essential for a successful analysis of KPIs, both at customer and provider end.

8.8 Conclusions

Firstly, we have carefully analyzed the specific characteristics of services. Then, we have studied how Business Intelligence can benefit from services at four different levels (i.e. IaaS, PaaS, SaaS and BaaS):

- In the first case, the power of Cloud Computing can help the costly BI applications.
- In the second, specific platforms (e.g., BigTable and MapReduce) have been developed to benefit from world scale parallelism in analytical tasks.
- In the third, market studies show the relevance of BI software as a service in the overall software as a service market.
- Finally, BPM and ETL modeling are clearly linked as part of a global design process.

On the other hand, we also outlined the importance of monitoring service business and showed how this can be done.

Acknowledgements. We would like to thank Ferran Sabaté for his help with CRM and SCM, as well as Victor López and Oscar Fernández for their insights and help with IBM products. Also a special acknowledge to the reviewers of the first version of the paper, for their help to improve the structure and readability of the paper.

This work has been partly supported by the Spanish Ministerio de Ciencia e Innovación under project TIN2008-03863.

References

1. Abadi, D.J.: Data management in the cloud: Limitations and opportunities. IEEE Data Engineering Bulletin 32(1), 3–12 (2009)
2. Aboulnaga, A., Salem, K., Soror, A.A., Minhas, U.F., Kokosielis, P., Kamath, S.: Deploying database appliances in the cloud. IEEE Data Eng. Bull. 32(1), 13–20 (2009)

3. Apache: Hadoop, http://hadoopapache.org/
4. Apache: HBase, http://hbaseapache.org/
5. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010)
6. Battré, D., Ewen, S., Hueske, F., Kao, O., Markl, V., Warneke, D.: Nephele/PACTs: A programming model and execution framework for web-scale analytical processing. In: Proceedings of the 1st ACM Symposium on Cloud computing (SoCC), pp. 119–130. ACM (2010)
7. Bowman, I.T., Bumbulis, P., Farrar, D., Goel, A.K., Lucier, B., Nica, A., Paulley, G.N., Smirnios, J., Young-Lai, M.: SQL Anywhere: An Embeddable DBMS. IEEE Data Engineering Bulletin 30(3), 29–36 (2007)
8. Castellanos, M., Simitsis, A., Wilkinson, K., Dayal, U.: Automating the loading of business process data warehouses. In: 12th International Conference on Extending Database Technology (EDBT), pp. 612–623. ACM (2009)
9. Chang, F., et al.: Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS) 26(2) (2008); preliminary version published in OSDI 2006
10. Curbera, F., Duftler, M.J., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing 6(2), 86–93 (2002)
11. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: 6th Symposium on Operating Systems Design and Implementation (OSDI), pp. 137–150 (2004)
12. DeWitt, D.J., Gray, J.: Parallel database systems: The future of high performance database systems. Commun. ACM 35(6), 85–98 (1992)
13. Díaz, D., Zaki, M., Theodoulidis, B., Sampaio, P.: A Systematic Framework for the Analysis and Development of Financial Market Monitoring Systems. In: SRII Global Conference (2011)
14. Engelbrecht, G., Bisbal, J., Benkner, S., Frangi, A.F.: Towards negotiable SLA-based QoS Support for Data Services. In: International Conference on Grid Computing (Grid), pp. 259–265 (2010)
15. Erl, T.: Service Oriented Architecture. Prentice Hall (2006)
16. Favre, C., Bentayeb, F., Boussaid, O.: Evolution of Data Warehouses' Optimization: A Workload Perspective. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 13–22. Springer, Heidelberg (2007)
17. Ferguson, D.F., Stockton, M.L.: Enterprise Business Process Management - Architecture, Technology and Standards. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 1–15. Springer, Heidelberg (2006)
18. Fitzsimmons, J., Fitzsimmons, M.: Service Management, 6th edn. McGraw-Hill (2008)
19. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query Recommendations for OLAP Discovery-Driven Analysis. IJDWM 7(2), 1–25 (2011)
20. Hostmann, B.: Business Intelligence as a Service: Findings and Recomendations. Research G00164653, Gartner (January 2009)
21. Inmon, W., Strauss, D., Neushloss, G.: DW 2.0. The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann (2008)
22. Isard, M.: Autopilot: automatic data center management. Operating Systems Review 41(2), 60–67 (2007)

23. Jiang, D., Ooi, B.C., Shi, L., Wu, S.: The performance of mapreduce: An in-depth study. PVLDB 3(1), 472–483 (2010)
24. Koutrika, G., Ioannidis, Y.E.: Personalization of queries in database systems. In: Proceedings of the 20th International Conference on Data Engineering (ICDE), pp. 597–608. IEEE Computer Society (2004)
25. Laudon, K.C., Laudon, J.P.: Management Information Systems: managing the digital firm. Prentice-Hall (2010)
26. Lovelock, C., Wright, L.: Services Marketing: People, Technology, Strategy, 6th edn. Prentice-Hall (2007)
27. Marotta, A., Piedrabuena, F., Abelló, A.: Managing Quality Properties in a ROLAP Environment. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 127–141. Springer, Heidelberg (2006)
28. Meijer, E., Bierman, G.: A Co-Relational Model of Data for Large Shared Data Banks. Communication of the ACM 54(4) (2011)
29. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Special Publication 800-145, National Institute of Standards and Technology (January 2011), draft
30. Melnik, S., et al.: Dremel: Interactive Analysis of Web-Scale Datasets. Proceedings of the VLDB Endowment (PVLDB) 3(1), 330–339 (2010)
31. zur Muehlen, M.: Volume versus variance: Implications of data-intensive workflows. IEEE Data Eng. Bull. 32(3), 42–47 (2009)
32. Özsu, T., Valduriez, P.: Principles of distributed database systems, 3rd edn. Prentice-Hall, Inc. (2011)
33. Papazoglou, M.P.: Web Services: Principles and Technology. Prentice Hall (2007)
34. Papazoglou, M., et al. (eds.): Service Research Challenges and Solutions for the Future Internet, vol. 6500. Springer, Heidelberg (2010)
35. Pavlo, A.: et al.: A comparison of approaches to large-scale data analysis. In: SIGMOD Int. Conf. on Management of Data, pp. 165–178. ACM (2009)
36. Pedersen, T.B.: Research challenges for cloud intelligence: invited talk. In: Proceedings of the 2010 EDBT/ICDT Workshops. ACM International Conference Proceeding Series (2010)
37. Ran, S.: A model for web services discovery with qos. SIGecom Exchanges 4(1), 1–10 (2003)
38. Romero, O., Abelló, A.: Automatic validation of requirements to support multidimensional design. Data Knowledge Engineering 69(9), 917–942 (2010)
39. Romero, O., Abelló, A.: A framework for multidimensional design of data warehouses from ontologies. Data Knowledge Engineering 69(11), 1138–1157 (2010)
40. Sampson, S.E.: Understanding Service Businesses: Applying Principles of Unified Services Theory, 2nd edn. John Wiley& Sons (2001)
41. Sánchez, L., García, F., Ruiz, F., Piattini, M.: Measurement in Business Processes: A Systematic Review. Business Process Management Journal 16(1), 114–134 (2010)
42. Stonebraker, M.: et al.: C-Store: A Column-oriented DBMS. In: 31st Int. Conf. on Very Large Data Bases (VLDB), pp. 553–564. ACM (2005)
43. Stonebraker, M., et al.: MapReduce and parallel DBMSs: friends or foes? Communication of ACM 53(1), 64–71 (2010)
44. Tien, J., Berg, D.: A case for service systems engineering. Journal of Systems Science and Systems Engineering 12(1), 13–38 (2003)

45. Truong, H.L., Dustdar, S.: Integrating data for business process management. IEEE Data Eng. Bull. 32(3), 48–53 (2009)
46. Tsangaris, M.M., Kakaletris, G., Kllapi, H., Papanikos, G., Pentaris, F., Polydoras, P., Sitaridi, E., Stoumpos, V., Ioannidis, Y.E.: Dataflow processing and optimization on grid and cloud infrastructures. IEEE Data Eng. Bull. 32(1), 67–74 (2009)
47. Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., Skiadopoulos, S.: A generic and customizable framework for the design of ETL scenarios. Information Systems 30(7), 492–525 (2005)
48. Vogels, W.: A Head in the Cloud - The Power of Infrastructure as a Service. In: First workshop on Cloud Computing and Applications, CCA (2008)
49. Weske, M.: Business Process Management. Springer, Heidelberg (2007)
50. You, J., Xi, J., Zhang, C., Guo, G.: HDW: A High Performance Large Scale Data Warehouse. In: International Multi-Symposium of Computer and Computational Sciences (IMSCCS), pp. 200–202. IEEE (2008)

9

Collaborative Business Intelligence

Stefano Rizzi

Department of Electronics, Computer Sciences and Systems (DEIS)
University of Bologna
Bologna, Italy
stefano.rizzi@unibo.it

Summary. The idea of collaborative BI is to extend the decision-making process beyond the company boundaries thanks to cooperation and data sharing with other companies and organizations. Unfortunately, traditional BI applications are aimed at serving individual companies, and they cannot operate over networks of companies characterized by an organizational, lexical, and semantic heterogeneity. In such distributed business scenarios, to maximize the effectiveness of monitoring and decision making processes there is a need for innovative approaches and architectures. Data warehouse integration is an enabling technique for collaborative BI, and has been investigated along three main directions: warehousing approaches, where the integrated data are physically materialized, federative approaches, where the integration is virtual and based on a global schema, and peer-to-peer approaches, that do not rely on a global schema to integrate the component data warehouses. In this paper we explore and compare these three directions by surveying the available work in the literature. Then we outline a new peer-to-peer framework, called Business Intelligence Network, where peers expose querying functionalities aimed at sharing business information for the decision-making process. The main features of this framework are decentralization, scalability, and full autonomy of peers.

Keywords: business intelligence, distributed databases, query reformulation, peer-to-peer architectures.

9.1 Introduction

A new generation of business intelligence (BI) systems has been emerging during the last few years to meet the new, sophisticated requirements of business users. The term *BI 2.0* has been coined to denote these systems; among their characterizing trends, we mention:

- *BI as a service*, where BI applications are hosted as a service provided to business users across the Internet.
- *Real-time BI*, where information about business operations is delivered as they occur, with near-0 latency.
- *Situational BI*, where information in an enterprise data warehouse is completed and enriched by correlating it with external information that may

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 186–205, 2012.

come from the corporate intranet, be acquired from some external vendor, or be derived from the internet.
- *Pervasive BI*, where information can be easily and timely accessed through devices with different computation and visualization capabilities, and with sophisticated and customizable presentations, by everyone in the organization.
- *Collaborative BI*, where a company information assets are empowered thanks to cooperation and data sharing with other companies and organizations, so that the decision-making process is extended beyond the company boundaries.

In particular, collaborative BI has been predicted to be the main BI trend for 2011 [1]. From the Wikipedia:

"Collaboration is working together to achieve a goal *[...]. It is a recursive process where two or more people or organizations work together to realize shared goals —this is more than the intersection of common goals, but a deep, collective, determination to reach an identical objective—* by sharing knowledge*, learning and building consensus. [...] Teams that work collaboratively can obtain* greater resources, recognition and reward *when facing competition for finite resources."*

Indeed, cooperation is seen by companies as one of the major means for increasing flexibility, competitiveness, and efficiency so as to survive in today uncertain and changing market. Companies need strategic information about the outer world, for instance about trading partners and related business areas [2]. Users need to access information anywhere it can be found, by locating it through a semantic process and performing integration on the fly. This is particularly relevant in inter-business collaborative contexts where companies organize and coordinate themselves to share opportunities, respecting their own autonomy and heterogeneity but pursuing a common goal.

Unfortunately, most information systems were devised for individual companies and for operating on internal information, and they give limited support to inter-company cooperation. In the same way, traditional BI applications are aimed at serving individual companies, and they cannot operate over networks of companies characterized by an organizational, lexical, and semantic heterogeneity. In such a complex and distributed business scenario, to maximize the effectiveness of monitoring and decision making processes there is a need for innovative approaches and architectures.

Data warehouse integration is an enabling technique for collaborative BI. It provides a broader base for decision-support and knowledge discovery than each single data warehouse could offer. Large corporations integrate their separately-developed departmental data warehouses; newly merged companies integrate their data warehouses into a central data warehouse;

autonomous but related organizations join together their data warehouses to enforce the decision making process [3].

Although the integration of heterogeneous databases has been widely discussed in the literature, only a few works are specifically focused on strategies for data warehouse integration. Two categories of approaches were mainly devised: *warehousing* approaches, where the integrated data are physically materialized, and *federative* approaches, where integration is virtual. In both cases, it is assumed that all components to be integrated share the same schema, or at least that a global schema is given. This assumption is perfectly reasonable in business contexts where a common view of the business is shared, or where one of the component parties has a clear leadership. In contexts where the different parties have a common interest in collaborating while fully preserving their autonomy and their view of business, defining a global schema is often unfeasible. To cope with this, the category of *peer-to-peer* (P2P) approaches, that do not rely on a global schema to integrate the component data warehouses, has been emerging during the last few years. In P2P approaches, each peer can formulate queries also involving the other peers, typically based on a set of mappings that establish semantic relationships between the peers' schemata.

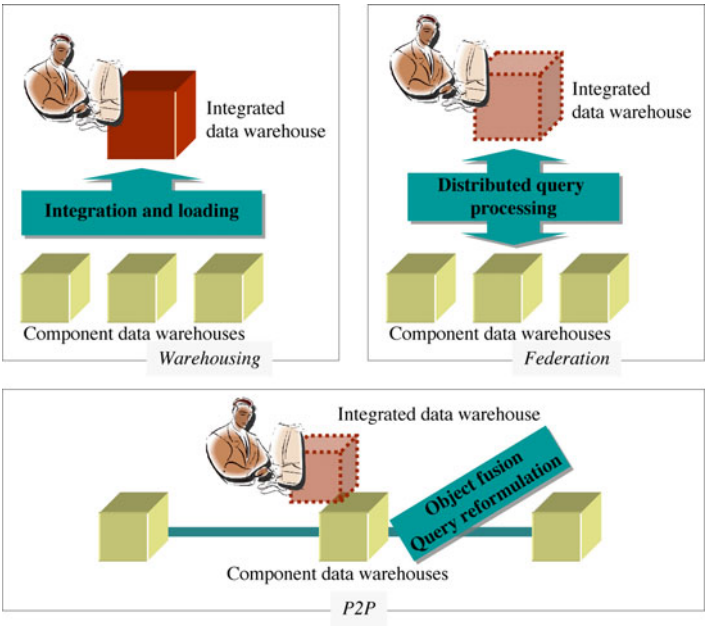


Fig. 9.1. Three approaches to collaborative BI

In this paper we compare these three categories by exploring the available works in the literature. In particular, after surveying the related literature

in the OLTP field in Section 9.2, in Sections 9.3, 9.4, and 9.5 we survey the warehousing, federative, and P2P approaches, respectively. Then in Section 9.6 we outline a new peer-to-peer framework, called Business Intelligence Network, where peers expose querying functionalities aimed at sharing business information for the decision-making process. The main features of this framework are decentralization, scalability, and full autonomy of peers. Finally, in Section 9.7 the conclusions are drawn.

9.2 Related Literature for OLTP

In the OLTP context, the research area sharing most similarities with warehousing approaches to collaborative BI is *data exchange*. In data exchange, data structured under one source schema must be restructured and translated into an instance of a different target schema, that is materialized [4]. In this scenario, the target schema is often independently created and comes with its own constraints that have to be satisfied.

On the other hand, federative approaches have their OLTP counterpart in *data integration systems*. Data from different sources are combined to give users a unified view [5]; in this way, users are freed from having to locate individual sources, learn their specific interaction details, and manually combine the data [6]. The unified view that reconciles the sources is represented by a global schema. In this case query processing requires a reformulation step: a query over the global, target schema has to be reformulated in terms of a set of queries over the sources.

Finally, P2P approaches to collaborative BI are related to the decentralized sharing of OLTP data between autonomous sources, that has been deeply studied in the context of *Peer Data Management Systems* (PDMSs). PDMSs were born as an evolution of mediator systems in the data integration field [7] and generalize data exchange settings [8]. A PDMS consists of a set of peers, each with an associated schema representing its domain of interest; peer mediation is implemented by means of semantic mappings between portions of schemata that are local to a pair or a small set of peers. Every peer can act freely on its data, and also access data stored by other peers without having to learn their schema and even without a mediated schema [9]. In a PDMS there is no a priori distinction between source and target, since a peer may simultaneously act as a distributor of data (thus, a source peer) and a recipient of data (thus, a target peer). As in the case of data integration systems, in a PDMS data remain at the sources and queries processing entails query reformulation over the peer schemata.

In all these contexts, modeling the relationships (*mappings*) between source and target schemata is a crucial aspect. Research in the data integration area has provided rich and well-understood schema mediation languages [5] to this end. The two commonly used formalisms are the *global-as-view* (GAV) approach, in which the mediated (global) schema is defined as a set of views over the data sources, and the *local-as-view* (LAV) approach, in which the

contents of data sources are described as views over the mediated schema. Depending on the kind of formalism adopted, GAV or LAV, queries posed to the system are answered differently, namely by means of query unfolding or query rewriting techniques [10], respectively. In a data exchange setting, assertions between a source query and a target query are used to specify what source data should appear in the target and how. These assertions can be represented neither in the LAV nor in the GAV formalisms, but rather they can be thought of as GLAV (*global-and-local-as-view*) [4]. A structural characterization of schema mapping languages is provided in [11], together with a list of the basic tasks that all languages ought to support.

In distributed OLTP environments, the schema mapping generation phase and the preceding schema matching phase pose new issues with reference to simpler centralized contexts: consistency problems are studied in [12] and innovative learning techniques are presented in [13]. Other studies in the field have focused on integrating the computation of core solutions in the mapping generation process, aimed at determining redundancy-free mappings in data exchange settings [14, 15].

Declaring useful mappings in the OLAP context necessarily requires also the level of instances to be taken into account. Unfortunately, in the OLTP literature the definition of mappings is typically done at the schema level, and the problem of managing differences in data formats has only marginally been considered. A seminal paper regarding this topic is [16], where constraint queries are translated across heterogeneous information sources taking into account differences in operators and data formats.

A related problem is that of reconciliation of results, that takes a primary role in federative and P2P approaches. In the OLTP context this issue is referred to as *object fusion* [17]. This involves grouping together information (from the same or different sources) about the same real-world entity. In doing this fusion, the mediator may also "refine" the information by removing redundancies, resolving inconsistencies between sources in favor of the most reliable source, and so on.

9.3 Warehousing Approaches

As already mentioned, in this family of approaches the data that result from the process of integrating a set of component data warehouses according to a global schema are materialized. The main drawback of these approaches is that they can hardly support dynamic scenarios like those of mergers and acquisitions.

An approach in this direction is the one proposed in [18]. Given two dimensions belonging to different data marts where a set of mappings between corresponding levels has been manually declared or automatically inferred, three properties (namely *coherence*, *soundness*, and *consistency*) that enable a compatibility check between the two dimensions are defined. A technique

that combines the contents of the dimensions to be integrated is then used to derive a materialized view that includes the component data marts.

A hybrid approach between the warehouse and the federation approach is suggested in [19] as a way to obtain a more flexible and applicable architecture. The idea is to aggregate selected data from the component data warehouses as materialized views and cache them at a federation server to improve query performance; a set of *materialized query tables* are recommended for the benefits of load distribution and easy maintenance of aggregated data.

Another borderline approach is proposed in [20]: while fact data are not physically integrated, a central *dimension repository* is used to replicate dimensional data (according to a global schema) from the component data warehouses, aimed at increasing querying efficiency. To effectively cope with evolutions in the schema of the components, a fact algebra and a dimension algebra are used in this approach for declaring maintainable mappings between the component schemata.

9.4 Federative Approaches

A *federated data warehouse*, sometimes also called *distributed data warehouse*, is a logical integration of data warehouses that provides transparent access to the component data warehouses across the different functions of an organization. This is achieved through a global schema that represents the common business model of the organization [21]. Differently from warehousing approaches, the integrated data are not physically stored, so queries formulated on the global schema must be rewritten on the component schemata. This adds complexity to the query management framework, but enables more flexible architectures where new component data warehouses can be dynamically inserted.

A distributed data warehouse architecture is outlined in [22], and a prototype named CubeStar for distributed processing of OLAP queries is introduced. CubeStar includes a middleware layer in charge of making the details of data distribution transparent to the front-end layer, by generating optimized distributed execution plans for user queries.

A distributed data warehouse architecture is considered also in [23] as a solution for contexts where the inherently distributed nature of the data collection process and the huge amount of data extracted make the adoption of a central repository impractical. The Skalla system for distributed query processing is proposed, with particular emphasis on techniques for optimizing both local processing and communication costs; however, since it is assumed that all collection points share the same schema, the approach cannot be used to cope with heterogeneous settings.

In the context of a federated architecture, with specific reference to the healthcare domain, the work in [24, 3] presents an algorithm for matching heterogeneous multidimensional structures, possibly characterized by different granularities for data. Mappings between the local schemata of the data

warehouses to be integrated and a given global schema are discovered in a semi-automated manner, based on a measure of similarity between complex concepts.

A process to build an integrated view of a set of data warehouses is outlined in [25]. This integrated view is defined as the largest common schema to all the components, and its instances are obtained by merging the instances of the components.

In [18], the problem of virtual integration of heterogeneous data marts is faced in a loosely-coupled scenario where there is a need for identifying the common information (intuitively, the intersection) between the components while preserving their autonomy. A set of rules to check for dimension compatibility are declared first, then drill-across queries are used to correlate on-the-fly the component data marts.

A *multi data warehouse* system is introduced in [26] as one relying on a distributed architecture where users are enabled to directly access the heterogeneous schemata of the component data warehouses, which makes the coupling between the components looser than in federated data warehouses. A SQL-MDi query language is proposed to transform a cube in order to make it compatible with a global, virtual cube and ready for integration. Specific attention is devoted to solving schema and instance conflicts among the different components.

An XML-based framework for supporting interoperability of heterogeneous data warehouses in a federation scenario is described in [27]. In the proposed architecture, a *federated layer* allows for restructuring and merging local data and schemas to provide a global, single view of the component data warehouses to the end users. XML is used both to represent the local data warehouse schemata, the global schema, and the mapping between them.

Another XML-based approach is the one in [28], that discusses the possible conflicts arising when heterogeneous data warehouses are integrated and proposes solutions to resolve the semantic discrepancies. Data cubes are transformed into XML documents and queried under a global view.

XML *topic maps* are used in [29] to integrate the information stored in distributed data warehouses. The schema integration process is based on merging local topic maps to generate global topic maps, taking different types of semantic conflicts into account.

A different approach is presented in [30], that introduces an architecture for *hierarchically distributed data warehouses* where component data warehouses are organized into a tree and data are progressively summarized level over level. A local OLAP query can be posed at any node of the tree, it is rewritten on remote nodes, and the results are merged.

9.5 Peer-to-Peer Approaches

Though federative approaches support more flexible and dynamic architectures than warehousing ones, still they do not fully preserve the autonomy

of individual actors. In complex business scenarios where no leadership can be established among a set of actors interested in cooperating, to maximize the effectiveness of monitoring and decision making processes there is a need for truly decentralized approaches. This can be achieved by relying on P2P architectures.

In [31, 32], the authors introduced the idea of using a P2P architecture for warehousing XML content. In their view, a P2P warehouse is not different from a centralized one from the logical point of view, while from the physical point of view information is distributed over a set of heterogeneous and autonomous peers rather than centralized. Because of this, query processing necessarily requires distributed computation. Among the advantages of this approach, we mention ownership (each peer has full control over its information) and dynamicity (peers can transparently enter and leave the system). How to map the local schema of each peer onto each other is one of the open problems.

The approach proposed in [33] reformulates XML queries over a set of peers hosting XML databases with heterogeneous (and possibly conflicting) schemata, in the absence of a global schema. Reformulation is based on mapping rules inferred from informal schema correspondences.

In [34, 35] the authors present a model for multidimensional data distributed across a P2P network, together with a mapping-based technique for rewriting OLAP queries over peers. In presence of conflicting dimension members, an approach based on belief revision is proposed to revise the instance of the source peer's dimension and adapt it to the instance of the target peer's dimension.

Another work centered on interoperability issues among heterogeneous data warehouses is the one by [36], that emphasizes the importance of a semantic layer to enable communication among different components. This approach supports the exchange of business calculation definitions and allows for their automatic linking to specific component data warehouses through semantic reasoning. Three models are suggested: a business ontology, a data warehouse ontology, and a mapping ontology between them.

As to performance aspects, in [37] the authors propose a P2P architecture for supporting OLAP queries focusing on the definition of a caching model to make the query rewriting process more efficient. They also define adaptive techniques that dynamically reconfigure the network structure in order to minimize the query cost.

Finally, as to the data reconciliation, a typical requirement in collaborative BI is the merging of results at different levels of aggregation. In this direction, the work proposed in [38] discusses a general approach on the use of aggregation operations in information fusion processes and suggests practical rules to be applied in common scenarios.

9.6 Business Intelligence Networks

In this section we describe a new framework to collaborative BI, called *Business Intelligence Network* (BIN), based on a P2P architecture [39]. BINs enable BI functionalities to be shared over networks of companies that, though they may operate in different geographical and business contexts, are chasing mutual advantages by acting in a conscious and agreed upon way. A BIN is based on a network of peers, one for each company participating in the consortium; peers are equipped with independent BI platforms that expose some functionalities aimed at sharing business information for the decision-making process, in order to create new knowledge (Figure 9.2). Remarkably, since each peer is allowed to define and change the set of shared information as well as its own terminology and schema without being subject to a shared schema, the BIN approach fully preserves peer autonomy. Besides, the BIN architecture is completely decentralized and scalable to cope with business contexts where the number of participants, the complexity of business models, and the user workload are unknown a priori and may change in time.

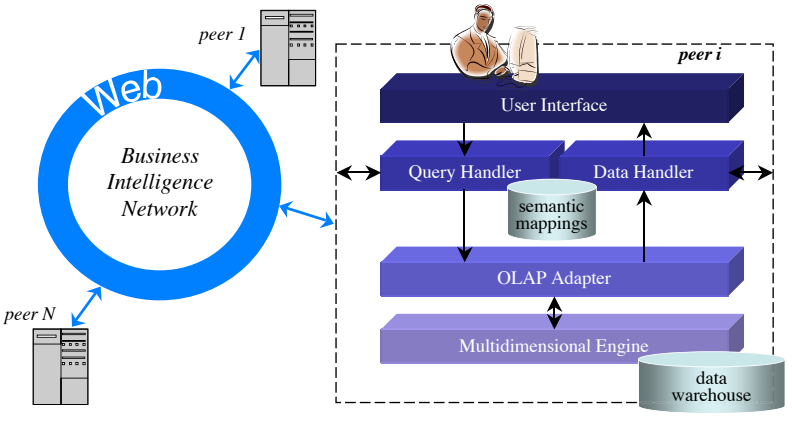


Fig. 9.2. A BIN architecture

The main benefits the BIN approach aims at delivering to the corporate world are (1) the possibility of building new inter-organizational relationships and coordination approaches, and (2) the ability to efficiently manage inter-company processes and safely share management information. Other practical benefits arising from activating a BIN depend on the specific corporate context; for instance, in companies that belong to the same supply-chain or operate in the same market, the (partial) business information sharing is required by users to allow inter-company processes to be monitored and markets to be accurately controlled.

The core idea of a BIN is that of enabling users to transparently access business information distributed over the network. A typical interaction sequence is the following:

1. A user formulates an OLAP query q by accessing the local multidimensional schema exposed by her peer, p.
2. Query q is processed locally on the data warehouse of p.
3. At the same time q is forwarded to the network.
4. Each involved peer locally processes the query on its data warehouse and returns its results to p.
5. The results are integrated and returned to the user.

The local multidimensional schemata of peers are typically heterogeneous. So, during distributed query processing, before a query issued on a peer can be forwarded to the network it must be first *reformulated* according to the multidimensional schemata of the source peers. Data are then extracted from each source peer and are mapped onto the schema of the querying peer, that plays the role of the target.

In line with the approach adopted in *Peer Data Management Systems* (PDMSs) [7], query reformulation in a BIN is based on *semantic mappings* that mediate between the different multidimensional schemata exposed by two peers, i.e., they describe how the concepts in the multidimensional schema of the target peer map onto those of the source peer. Direct mappings cannot be realistically defined for all the possible couples of peers. So, to enhance information sharing, a query q issued on p is forwarded to the network by first sending it to (a subset of) the immediate neighbors of p, then to their immediate neighbors, and so on. In this way, q undergoes a chain of reformulations along the peers it reaches, and results are collected from any peer that is connected to p through a path of semantic mappings. This process is sketched in Figure 9.3.

The approach outlined above is reflected by the internal architecture of each peer, sketched in the right side of Figure 9.2, whose components are:

1. *User Interface.* A web-based component that manages bidirectional interaction with users, who use it to visually formulate OLAP queries on the local multidimensional schema and explore query results.
2. *Query Handler.* This component receives an OLAP query from either the user interface or a neighboring peer on the network, sends that query to the OLAP adapter to have it locally answered, reformulates it onto the neighboring peers (using the available semantic mappings), and transmits it to those peers.
3. *Data Handler.* When the peer is processing a query that was locally formulated (i.e., it is acting as a target peer), the data handler collects query results from the OLAP adapter and from the source peers, integrates them, and returns them to the user interface. When the peer is processing a query that was formulated on some other peer p (i.e., it is

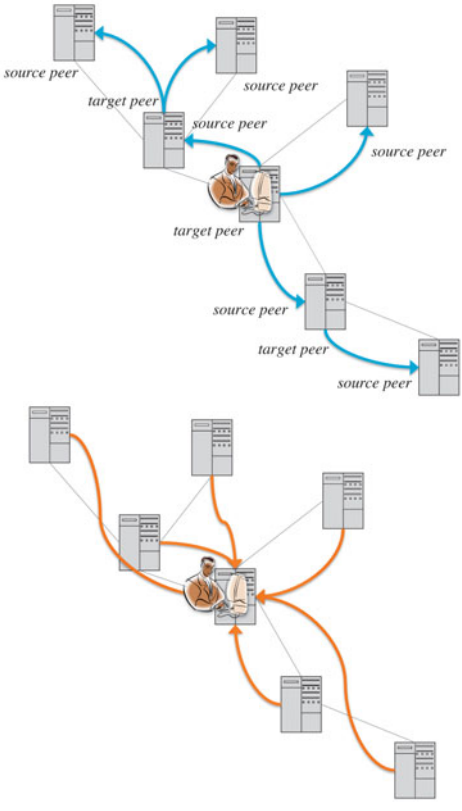


Fig. 9.3. Recursive query reformulation (left) and collection of results (right)

acting as a source peer), the data handler just collects local query results from the OLAP adapter and returns them to the target peer p.

4. *OLAP Adapter*. This component adapts queries received from the query handler to the querying interface exposed by the local multidimensional engine.
5. *Multidimensional Engine*. It manages the local data warehouse according to the multidimensional schema representing the peer's view of the business, and provides MDX-like query answering functionalities.

Interactions between peers are based on a message-passing protocol.

9.6.1 Mapping Language

Reformulation of OLAP queries first of all requires a language for properly expressing the *semantic mappings* between each couple of neighboring peers;

this language must accommodate the peculiar characteristics of the multidimensional model, on which the representation of business information at each peer is founded. The basic requirements for the mapping language are:

1. The asymmetry between dimensions and measures should be reflected in the mapping language by providing different predicates for mapping dimensions/attributes and measures.
2. Since aggregation is inherently part of the multidimensional model, the mapping language should be capable of specifying the relationship between two attributes of different multidimensional schemata in terms of their granularity.
3. A measure is inherently associated with an aggregation operator. Then, when mapping a measure onto another, their aggregation operators must be taken into account to avoid the risk of inconsistent query reformulations.
4. Declaring useful mappings in the BI context necessarily requires also the instance level to be taken into account. This can be done if there is a known way to transcode values of an attribute/measure belonging to a multidimensional schema into values of an attribute/measure belonging to another multidimensional schema.

The language used in a BIN to express how the multidimensional schema $\mathcal{M}_s$ of a source peer s maps onto the multidimensional schema $\mathcal{M}_t$ of a target peer t includes five *mapping predicates*, that will be explained below. In general, a mapping establishes a semantic relationship from one or more concepts (either measures or attributes) of $\mathcal{M}_s$ to one or more concepts of $\mathcal{M}_t$, and enables a BIN query formulated on $\mathcal{M}_t$ to be reformulated on $\mathcal{M}_s$. Optionally, a mapping involving attributes can be annotated with a *transcoding function* that specifies how values of the target concepts can be obtained from values of the source concepts. If this function is available, it is used to increase the reformulation effectiveness.

- `same` predicate: μ_t `same`$_{expr}$ M_s, where $\mu_t = \langle m_t, \alpha_t \rangle$ is a metric[1] of $\mathcal{M}_t$, M_s is a subset of measures of $\mathcal{M}_s$, and *expr* is an expression involving the measures in M_s. This mapping predicate is used to state that whenever m_t is asked in a query on $\mathcal{M}_t$ using α_t, it can be rewritten as *expr* on $\mathcal{M}_s$.
- `equi-level` predicate: P_t `equi-level`$_f$ P_s, where P_t and P_s are sets of attributes of $\mathcal{M}_t$ and $\mathcal{M}_s$, respectively. This predicate is used to state that P_t has the same semantics and granularity as P_s. Optionally, it can be annotated with an injective transcoding $f : Dom(P_s) \rightarrow Dom(P_t)$ that establishes a one-to-one relation between tuples of values of P_s and P_t, and is used to integrate data returned by the source and target peers.

[1] A *metric* of a multidimensional schema $\mathcal{M}$ is a couple $\mu = \langle m, \alpha \rangle$, where m is a measure and α is a valid aggregation operator for m.

- `roll-up` predicate: P_t `roll-up`$_f$ P_s. This predicate states that P_t is a roll-up of (i.e., it aggregates) P_s. Optionally, it can be annotated with a non-injective transcoding $f : Dom(P_s) \rightarrow Dom(P_t)$ that establishes a many-to-one relation between tuples of values of P_s and P_t, and is used to aggregate data returned by the source peer and integrate them with data returned by the target peer.
- `drill-down` predicate: P_t `drill-down`$_f$ P_s. This predicate is used to state that P_t is a drill-down of (i.e., it disaggregates) P_s. Optionally, it can be annotated with a non-injective transcoding $f : Dom(P_t) \rightarrow Dom(P_s)$ that establishes a one-to-many relation between tuples of values of P_s and P_t. The transcoding f cannot be used to integrate data returned by t and s because this would require disaggregating data returned by s, which obviously cannot be done univocally; however, it can be used to reformulate the selection predicates expressed at t onto s.
- `related` predicate: P_t `related` P_s. This predicate is used to state that P_t and P_s tuples of values have a many-to-many relationship.

Example 1. As a working example, we consider a BIN for sharing information about funded research projects among European nations. Figure 9.4 shows the multidimensional schemata of related facts at the peers in London and Rome, using the Dimensional Fact Model notation [40]; small circles represent attributes, while measures are listed inside the fact boxes. Note that, while an event in the Rome schema corresponds to the funding given to each research unit within a project, in the London schema an event aggregates the projects by their coordinator. Figure 9.4 also shows some of the mappings that can be

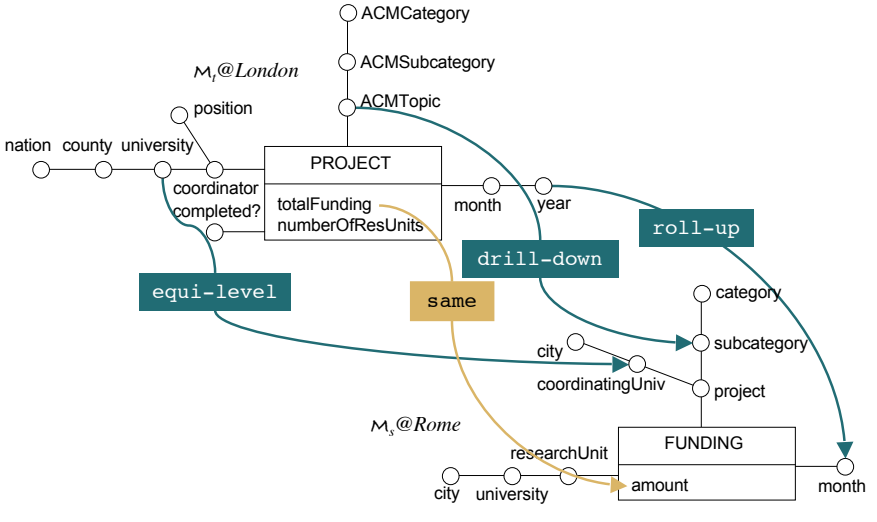


Fig. 9.4. Multidimensional schemata of related facts at two peers

defined to reformulate queries expressed in London (target peer) according to the schema adopted in Rome (source peer). As examples of transcodings, consider the function that associates each topic in the ACM classification with a subcategory and the one that associates each month with its year, used to annotate mappings ACMTopic `drill-down` subcategory and year `roll-up` month, respectively. Similarly, the `same` mapping between totalFunding and amount is annotated with an expression that converts euros into pounds.

9.6.2 Query Reformulation

Reformulation takes as input an OLAP query on a target schema $\mathcal{M}_t$ as well as the mappings between $\mathcal{M}_t$ and the schema of one of its neighbor peers, the source schema $\mathcal{M}_s$, to output an OLAP query that refers only to $\mathcal{M}_s$. The reformulation framework we propose is based on a relational setting, as depicted in Figure 9.5, where the multidimensional schemata, OLAP queries, and semantic mappings at the OLAP level are translated to the relational model. As to multidimensional schemata, without loss of generality we assume that they are stored at the relational level as star schemata. As to queries, a classic logic-based syntax is adopted to express them at the relational level. As to mappings, their representation at the relational level uses a logical formalism typically adopted for schema mapping languages, i.e., *source-to-target tuple generating dependencies* (s-t tgd's) [11]. A query is then reformulated starting from its relational form on a star schema, using the mappings expressed as s-t tgd's.

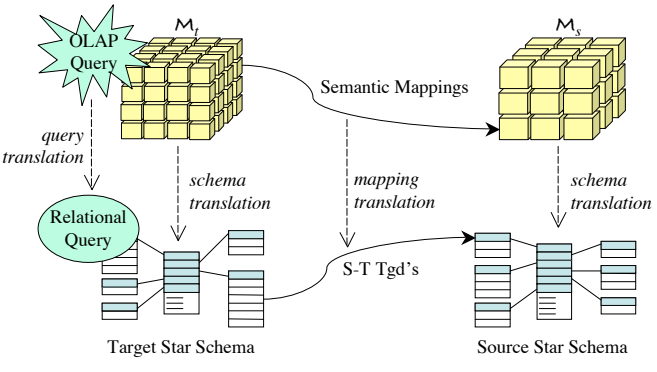


Fig. 9.5. A reformulation framework

A detailed explanation of the reformulation process can be found in [41]. Here we just provide an intuition based on a couple of examples. Note that, remarkably, all the translation steps outlined below are automated; the only manual phase in the whole process is the definition of the semantic mappings between each couple of neighboring peers.

Example 2. To start simple, consider the OLAP query q asking, at the London peer, for the total funding of projects about each subcategory of category 'Information Systems' on April 2011. Consistently with the framework sketched in Figure 9.5, reformulating this query onto the Rome peer requires:

1. Translating the multidimensional schemata at both London and Rome into star schemata, which produces the result shown in Figure 9.6.

@London
ProjectFT(coordinator,completed,ACMTopic,month,totalFunding,numberOfResUnits)
CoordinatorDT(coordinator,university,county,nation,position)
ACMTopicDT(ACMTopic,ACMSubcategory,ACMCategory)
MonthDT(month,year)

@Rome
FundingFT(researchUnit,project,month,amount)
ResearchUnitDT(researchUnit,university,city)
ProjectDT(project,subcategory,category,coordinatingUniv,city)

Fig. 9.6. Star schemata for the London and Rome peers

2. Translating q into a relational query on the London star schema:

$$q : \pi_{\mathsf{ACMSubcategory},\mathtt{SUM}(\mathsf{totalFunding})}$$
$$\sigma_{(\mathsf{month}=\text{'April 2011'},\mathsf{ACMCategory}=\text{'Inf. Sys.'})}\chi_{London}$$

   where χ_{London} denotes the star join made over the London star schema.
3. Translating the mappings involved into s-t tgd's. For this query, the involved mappings are:

$$\mathsf{ACMCategory}\ \mathtt{equi\text{-}level}_f\ \mathsf{category}$$
$$\mathsf{ACMSubcategory}\ \mathtt{equi\text{-}level}_g\ \mathsf{subcategory}$$
$$\mathsf{month}\ \mathtt{equi\text{-}level}_h\ \mathsf{month}$$
$$\langle \mathsf{totalFunding}, \mathtt{SUM}\rangle\ \mathtt{same}_{expr}\ \mathsf{amount}$$

   where f and g are the identity function, h converts the Rome format for months ('04-11') into the London format ('April 2011'), and *expr* is $\mathsf{amount} * 0.872$.

Using the reformulation algorithm proposed in [41], q is then translated into the following query over the Rome schema:

$$q' : \pi_{\mathsf{subcategory},\mathtt{SUM}(\mathsf{amount}*0.872)}\sigma_{(h(\mathsf{month})=\text{'April 2011'},\mathsf{category}=\text{'Inf. Sys.'})}\chi_{Rome}$$

Remarkably, in this case reformulation is *compatible*, i.e., it fully preserves the semantics of q. When a compatible reformulation is used, the results returned by the source peer do *exactly* match with q so they can be seamlessly integrated with those returned by the target peer.

Example 3. Consider now the query asking, at the London peer, for the yearly funding of projects about topic 'Heterogeneous Databases':

$$q : \pi_{\mathsf{year},\mathtt{SUM}(\mathsf{totalFunding})}\sigma_{(\mathsf{ACMTopic}=\text{'Heterogeneous Databases'})}\chi_{London}$$

In this case the mappings involved in reformulation are

$$\begin{aligned} \mathsf{ACMTopic}\ \mathtt{drill\text{-}down}_r\ \mathsf{subcategory} \\ \mathsf{year}\ \mathtt{roll\text{-}up}_s\ \mathsf{month} \\ \langle \mathsf{totalFunding}, \mathtt{SUM} \rangle\ \mathtt{same}_{expr}\ \mathsf{amount} \end{aligned}$$

where r is a function that associates each topic with its subcategory in the ACM classification, and s associates each month with its year. Query q is then translated into the following query over the Rome schema:

$$q' : \pi_{s(\mathsf{month}),\mathtt{SUM}(\mathsf{amount}*0.872)}\sigma_{(\mathsf{subcategory}=r(\text{'Heterogeneous Databases'}))}\chi_{Rome}$$

Differently from Example 2, here reformulation is not compatible, so the results returned by the Rome peer match q with some approximation. In particular, since the topic detail is not present in the Rome schema, data are returned for the whole subcategory of 'Heterogeneous Databases' (i.e., 'DATABASE MANAGEMENT'). Although an integration with the London data is not possible in this case, users can still exploit the results coming from Rome, for example by comparing them with the London data at the finest common aggregation level (subcategory in this case).

See [41] for a discussion of the issues arising with compatible and non compatible reformulations.

9.6.3 Open Issues

A BIN is a complex system, whose construction and management requires sophisticated techniques, mostly devised to cope with the peculiarities of decision-making oriented information. In the following we outline the main issues to be solved to ensure that a BIN operates in a reliable, effective, and efficient way [42]:

- Answering queries in a BIN may be a very resource-consuming task both for the computational effort which is required to each queried peer and for the amount of exchanged messages. In order to avoid this, techniques for optimizing the reformulation process in the network must be adopted. In particular, *query routing strategies* should be used to prune redundant paths and forward queries to the most promising peers only, and distributed caching models should be together with online aggregation techniques to minimize query execution costs.

- A BIN must provide a unified, integrated vision of the heterogeneous information collected from the different peers to answer a user query. To this end, *object fusion* functionalities must be adopted to properly reconcile the multidimensional results returned; this task is made more complex by the fact that, due to heterogeneity of multidimensional schemata, the information returned may be not completely compliant with the original user query (e.g., it may have a different granularity).
- When a user performs a query, the other peers will often return results that are not exactly conformed to the schema of that query. For this reason, a BIN requires *smart user interfaces* capable of emphasizing the differences and relationships between the returned data, as well as techniques to rank the returned data depending on how compliant they are with the original local query.
- A BIN should include mechanisms for controlling *data provenance and quality* in order to provide users with information they can rely on. A mechanism for data lineage is also necessary to help users understand the semantics of the retrieved data and how these data have been transformed to handle heterogeneity.
- The nature of the exchanged information, as well as the presence of participants that belong to different organizations, require advanced approaches for *security*, ranging from proper access policies to data sharing policies that depend on the degree of trust between participants, as well as techniques for protecting against undesired information inference.

9.7 Conclusion

Extending the existing BI architectures and techniques to support collaborative scenarios is a challenging task that lays the foundations for BI 2.0. In this paper we have presented the benefits of collaborative BI and we have discussed the different approaches in the literature. Then we have outlined the BIN approach, that uses a P2P architecture to enable decentralized information sharing across a network of heterogeneous and autonomous peers.

As shown in Section 9.6.3, several interesting research directions can be explored to enrich and improve the BIN approach. Though most of them are strictly related to research issues already tackled for OLTP data in P2P systems, the solutions presented in that context do not adequately apply to the specific scenario of a BIN, because they do not effectively deal with the peculiarities of multidimensional data and OLAP query processing. For instance, the BIN mapping language could be effectively coupled with a language for expressing user preferences, aimed at better tailoring the reformulation process to the user's wishes. As a first step in this direction, the specification of mappings can be enriched with a similarity score to express the semantic correlation of the source and target sets of concepts. Similarity may depend on the differences in the peers' vocabularies as well as on different perspectives of data representation (e.g., different granularities), and should be influenced

by the absence or presence of transcoding functions that make source and target values compatible. Such a score, representative of the semantic strength of a mapping, can be profitably employed in query reformulation where, in presence of alternative mappings for a given set of concepts onto a source peer, it can be used to identify the mapping that best approximates this set of concepts, so as to translate the query as accurately as possible. Then, the similarity scores of the mappings involved in the reformulation process can be combined to determine how compliant the results obtained from a source peer are, overall, with respect to the original query [3]. This compliance score can be profitably used by users to rank (and, possibly, filter) the results obtained from different source peers.

References

1. Lachlan, J.: Top 13 business intelligence trends for (2011), http://www.japan.yellowfin.bi
2. Hoang, T.A.D., Nguyen, T.B.: State of the art and emerging rule-driven perspectives towards service-based business process interoperability. In: Proc. Int. Conf. on Comp, and Comm. Tech., Danang City, Vietnam, pp. 1–4 (2009)
3. Banek, M., Vrdoljak, B., Tjoa, A.M., Skocir, Z.: Automated integration of heterogeneous data warehouse schemas. IJDWM 4(4), 1–21 (2008)
4. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. In: Proc. ICDT, pp. 207–224 (2003)
5. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. PODS, pp. 233–246 (2002)
6. Halevy, A.Y.: Technical perspective - schema mappings: rules for mixing data. Commun. ACM 53(1) (2010)
7. Halevy, A.Y., Ives, Z.G., Madhavan, J., Mork, P., Suciu, D., Tatarinov, I.: The Piazza peer data management system. IEEE TKDE 16(7), 787–798 (2004)
8. Fuxman, A., Kolaitis, P.G., Miller, R.J., Tan, W.C.: Peer data exchange. In: Proc. PODS, pp. 160–171 (2005)
9. Tatarinov, I., Halevy, A.Y.: Efficient query reformulation in peer-data management systems. In: Proc. SIGMOD Conf., Paris, France, pp. 539–550 (2004)
10. Halevy, A.: Answering queries using views: A survey. VLDB Journal 10(4), 270–294 (2001)
11. ten Cate, B., Kolaitis, P.G.: Structural characterizations of schema-mapping languages. Commun. ACM 53(1), 101–110 (2010)
12. Cudré-Mauroux, P., Aberer, K., Feher, A.: Probabilistic message passing in peer data management systems. In: Proc. ICDE, Atlanta, USA, p. 41 (2006)
13. Madhavan, J., Bernstein, P.A., Doan, A., Halevy, A.Y.: Corpus-based schema matching. In: Proc. ICDE, Tokyo, Japan, pp. 57–68 (2005)
14. Mecca, G., Papotti, P., Raunich, S.: Core schema mappings. In: Proc. SIGMOD, pp. 655–668 (2009)
15. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: getting to the core. ACM Trans. Database Syst. 30(1), 174–210 (2005)
16. Chang, K.C., Garcia-Molina, H.: Mind your vocabulary: Query mapping across heterogeneous information sources. In: Proc. SIGMOD, pp. 335–346 (1999)

17. Papakonstantinou, Y., Abiteboul, S., Garcia-Molina, H.: Object fusion in mediator systems. In: Proc. VLDB, Bombay, India, pp. 413–424 (1996)
18. Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. Distributed and Parallel Databases 23(1), 69–97 (2008)
19. Jiang, H., Gao, D., Li, W.S.: Exploiting correlation and parallelism of materialized-view recommendation for distributed data warehouses. In: Proc. ICDE, Istanbul, Turkey, pp. 276–285 (2007)
20. Berger, S., Schrefl, M.: From federated databases to a federated data warehouse system. In: Proc. HICSS, Waikoloa, Big Island of Hawaii, p. 394 (2008)
21. Jindal, R., Acharya, A.: Federated data warehouse architecture (2004), http://www.wipro.com/
22. Albrecht, J., Lehner, W.: On-line analytical processing in distributed data warehouses. In: Proc. IDEAS, pp. 78–85 (1998)
23. Akinde, M.O., Böhlen, M.H., Johnson, T., Lakshmanan, L.V.S., Srivastava, D.: Efficient OLAP query processing in distributed data warehouses. Inf. Syst. 28(1-2), 111–135 (2003)
24. Banek, M., Tjoa, A.M., Stolba, N.: Integrating Different Grain Levels in a Medical Data Warehouse Federation. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 185–194. Springer, Heidelberg (2006)
25. Schneider, M.: Integrated vision of federated data warehouses. In: Proc. DISWEB, Luxemburg (2006)
26. Berger, S., Schrefl, M.: Analysing Multi-Dimensional Data across Autonomous Data Warehouses. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 120–133. Springer, Heidelberg (2006)
27. Mangisengi, O., Huber, J., Hawel, C., Eßmayr, W.: A Framework for Supporting Interoperability of Data Warehouse Islands using XML. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2001. LNCS, vol. 2114, pp. 328–338. Springer, Heidelberg (2001)
28. Tseng, F.S.C., Chen, C.W.: Integrating heterogeneous data warehouses using XML technologies. J. Information Science 31(3), 209–229 (2005)
29. Bruckner, R.M., Ling, T.W., Mangisengi, O., Tjoa, A.M.: A framework for a multidimensional OLAP model using topic maps. In: Proc. WISE (2), pp. 109–118 (2001)
30. Zhou, S., Zhou, A., Tao, X., Hu, Y.: Hierarchically distributed data warehouse. In: Proc. HPC, Washington, DC, pp. 848–853 (2000)
31. Abiteboul, S.: Managing an XML Warehouse in a P2P Context. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 4–13. Springer, Heidelberg (2003)
32. Abiteboul, S., Manolescu, I., Preda, N.: Constructing and querying peer-to-peer warehouses of XML resources. In: Proc. SWDB, Toronto, Canada, pp. 219–225 (2004)
33. Bonifati, A., Chang, E.Q., Ho, T., Lakshmanan, L.V.S., Pottinger, R., Chung, Y.: Schema mapping and query translation in heterogeneous P2P XML databases. VLDB J 19(2), 231–256 (2010)
34. Espil, M.M., Vaisman, A.A.: Aggregate queries in peer-to-peer OLAP. In: DOLAP, Washington, DC, USA, pp. 102–111 (2004)
35. Vaisman, A., Espil, M.M., Paradela, M.: P2P OLAP: Data model, implementation and case study. Information Systems 34(2), 231–257 (2009)

36. Kehlenbeck, M., Breitner, M.H.: Ontology-Based Exchange and Immediate Application of Business Calculation Definitions for Online Analytical Processing. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 298–311. Springer, Heidelberg (2009)
37. Kalnis, P., Ng, W.S., Ooi, B.C., Papadias, D., Tan, K.L.: An adaptive peer-to-peer network for distributed caching of OLAP results. In: Proc. SIGMOD, Madison, Wisconsin, pp. 25–36 (2002)
38. Dubois, D., Prade, H.: On the use of aggregation operations in information fusion processes. Fuzzy Sets and Systems 142(1) (2004)
39. Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: Towards OLAP query reformulation in peer-to-peer data warehousing. In: Proc. DOLAP, pp. 37–44 (2010)
40. Golfarelli, M., Rizzi, S.: Data Warehouse design: Modern principles and methodologies. McGraw-Hill (2009)
41. Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: OLAP query reformulation in peer-to-peer data warehousing. Information Systems (to appear, 2011)
42. Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: BIN: Business intelligence networks. In: Business Intelligence Applications and the Web: Models, Systems and Technologies. IGI Global (2011)

Author Index